Wright State University

Report-P6

Paul Fuchs

U00747698

CEG 3900 Mobile and Cloud Computing

Prabhaker Mateti

16 April 2017

https://github.com/helmhammerhand/ceg-3900-spring-2017/

Table of Contents

Status of Project:

3.1 John
I only got the guest password (guest007)

Deliverables: Screenshots in this document

3.2 Hashcat:
It took me awhile to get hashcat and opencl installed and running. In the end, I was only able to run it on GPU but not CPU. Note: My APK used port 80 instead of ssh. This made it easier to convert to running on the cloud

Deliverables: Screenshots in this document, APK and server (same code as 3.3)

3.3 Haschat APK:
I used a docker image of hashcat to get around installation issues.

Deliverables: Screenshots, android project/APK (on github), Java server (on github)

3.4 Docker:
I built a container and from it ran the painbow example migrate, encrypt, and decrypt commands as shown on painbow's githyb page

Deliverables: Screenshots, Dockerfile used to build the container (on github)

3.5 Password Security APK
Contains the password strength estimator and word list comparer (note the word list server is not currently running). Also contains links to educational articles on password security.

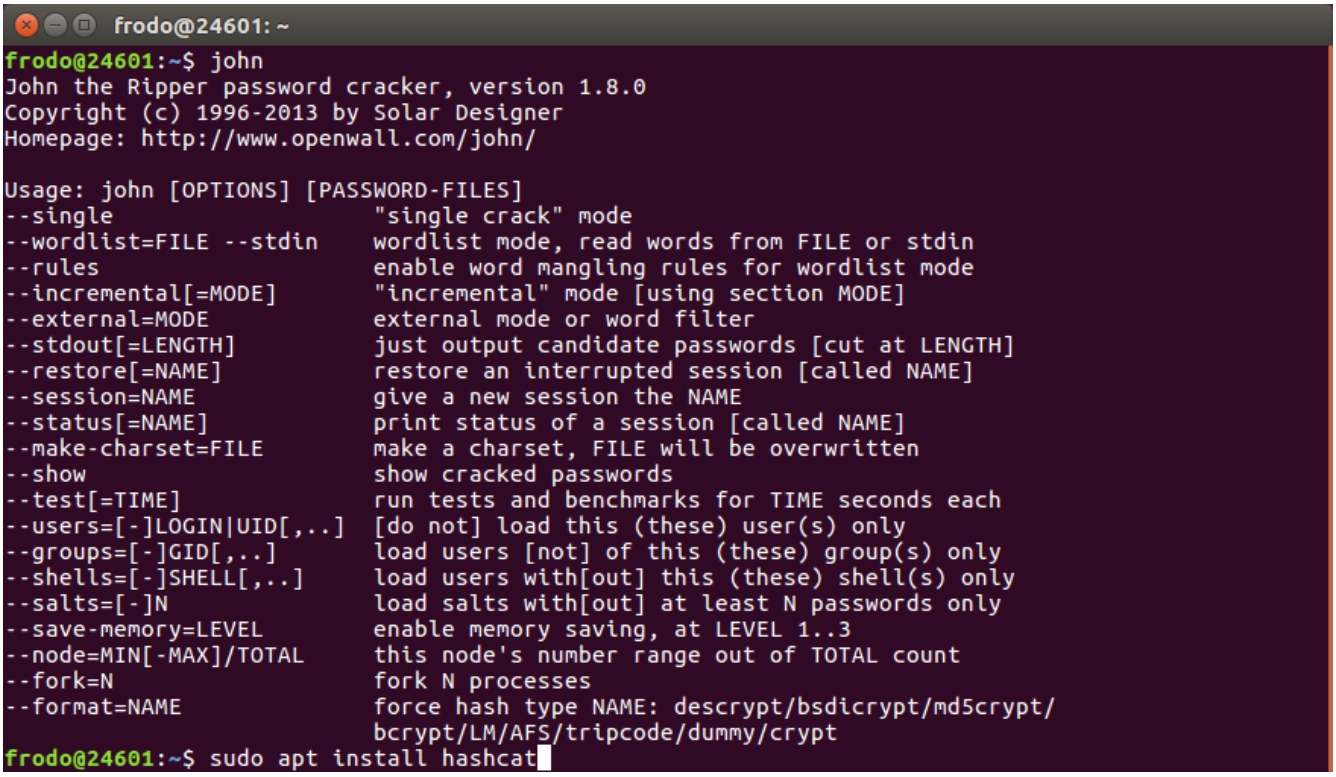
Deliverables: Android project/APK (on github)

Thoughts:

This project did take a good amount of time but went much faster than P5. Getting hashcat to run was surprisingly hard and I only ever got the GPU version working. It could be useful to provide some information on how to install drivers for future classes as this is where I had the most trouble (at one point my graphics driver got messed up and I had to boot to TTY to fix it).

3.1 John (1 hour)

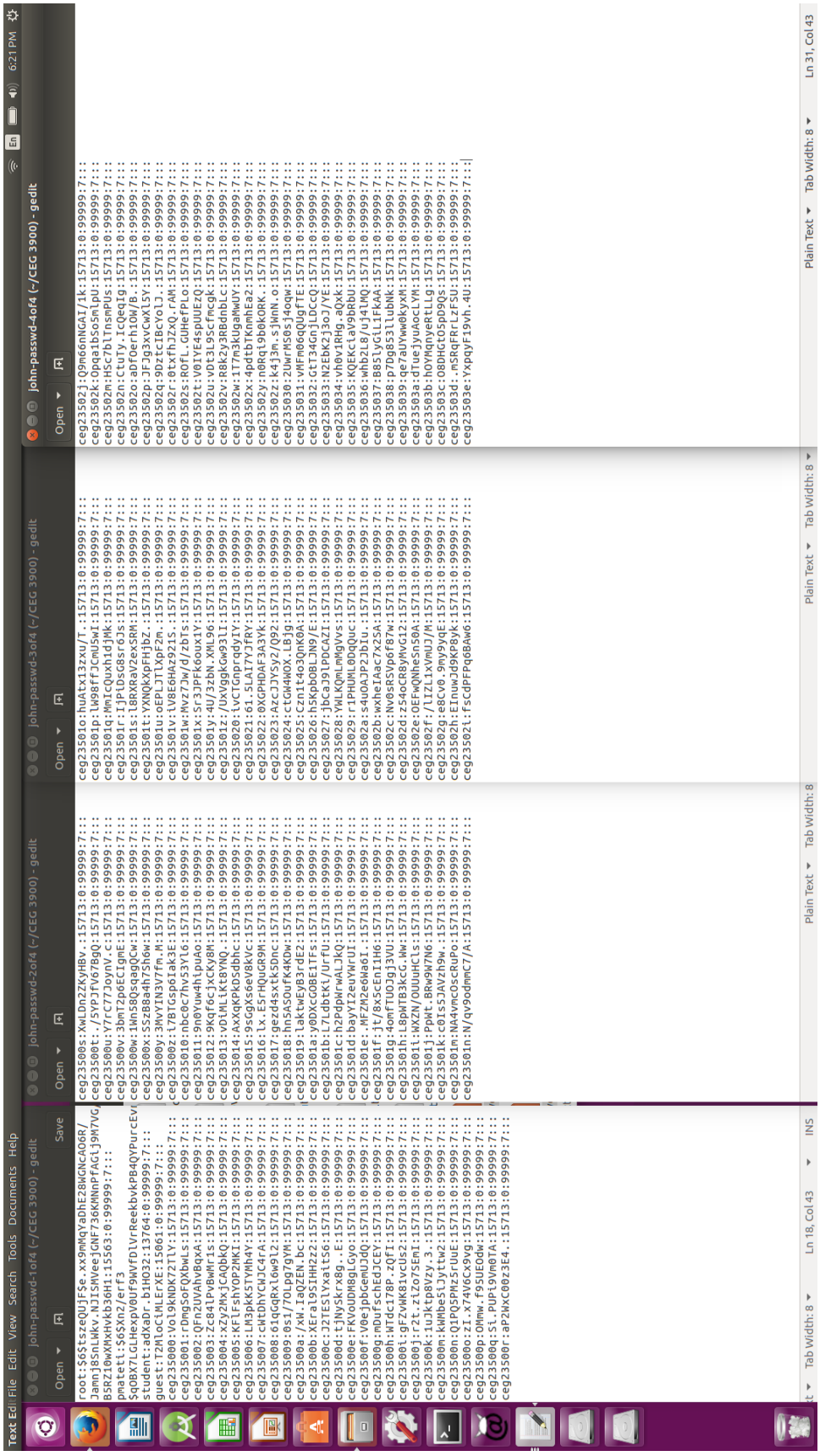
Installation



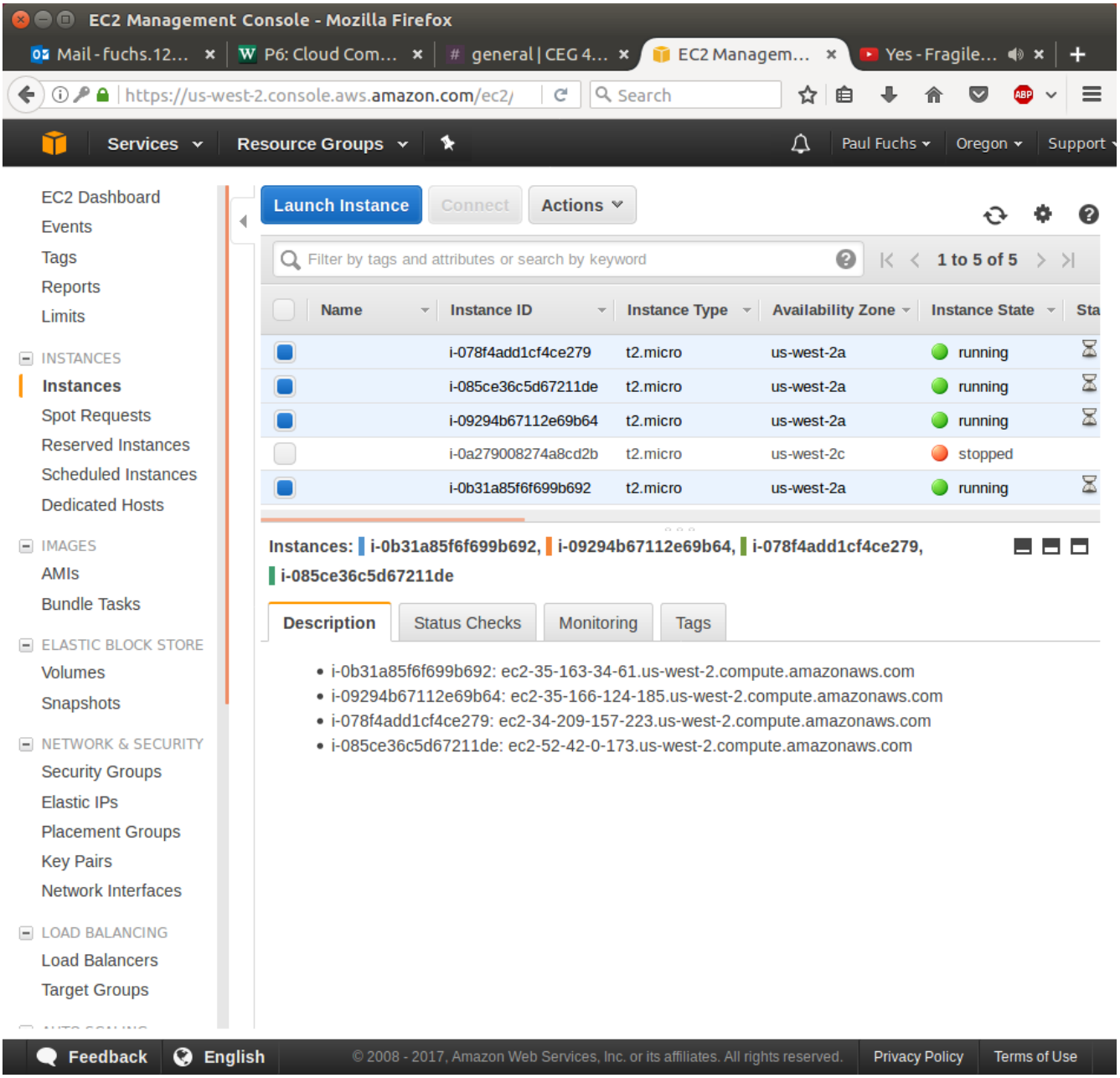
Running on my local machine, I got the password guest007.

John on the Cloud

I divided the password list into four files, each containing about thirty hashes.
The files are named john-passwd-nof4 where n is the file's number (1 through 4)

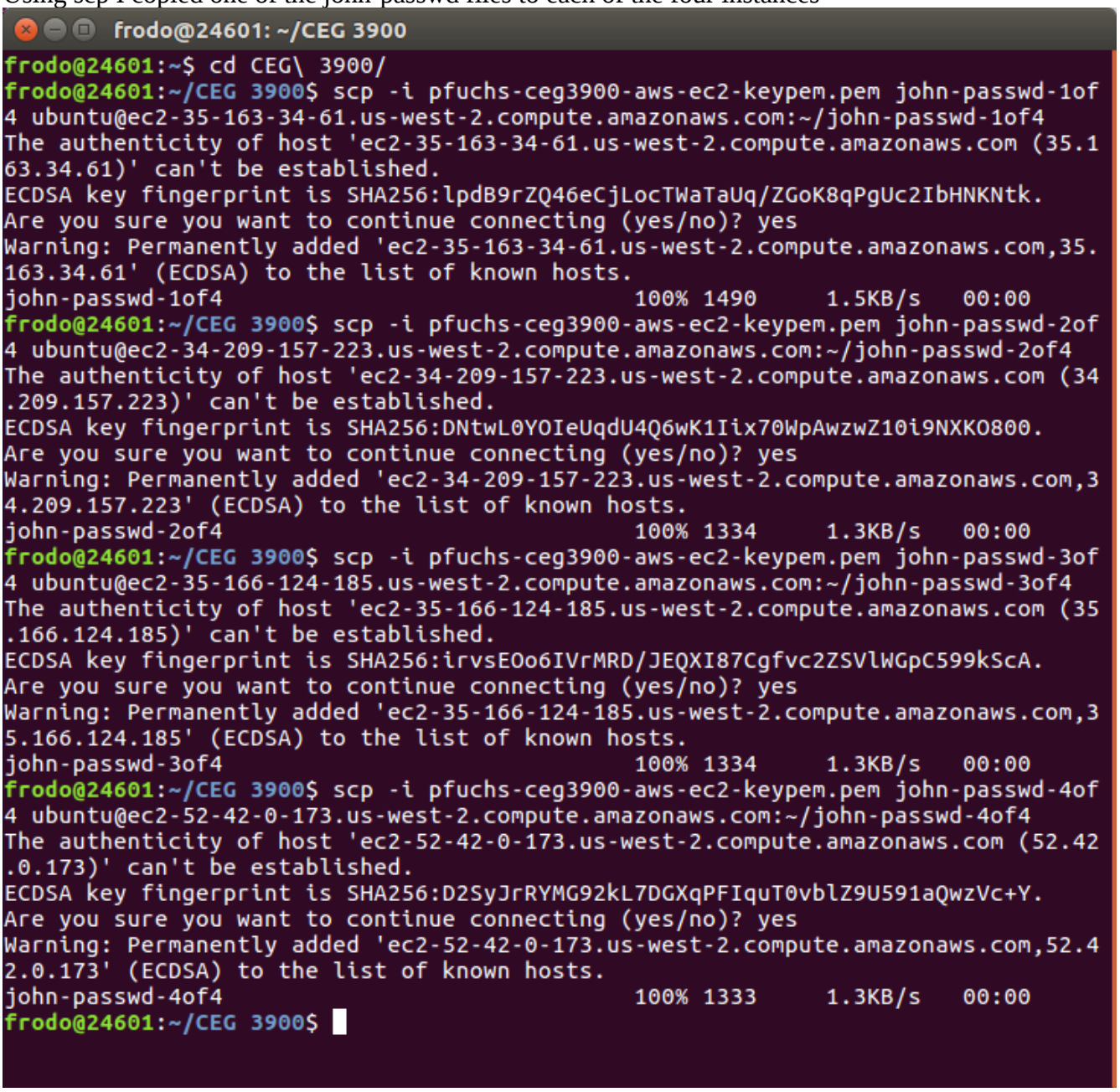


Running 4 instances on AWS
The instances I created and their domain names are shown below.



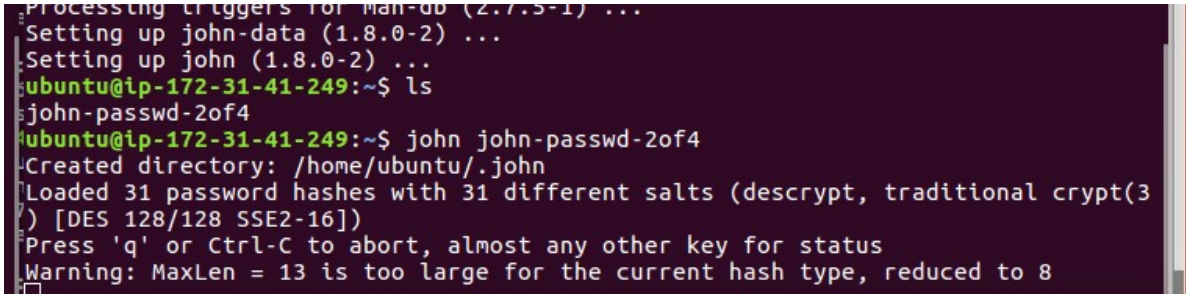
7

Using scp I copied one of the john-passwd files to each of the four instances

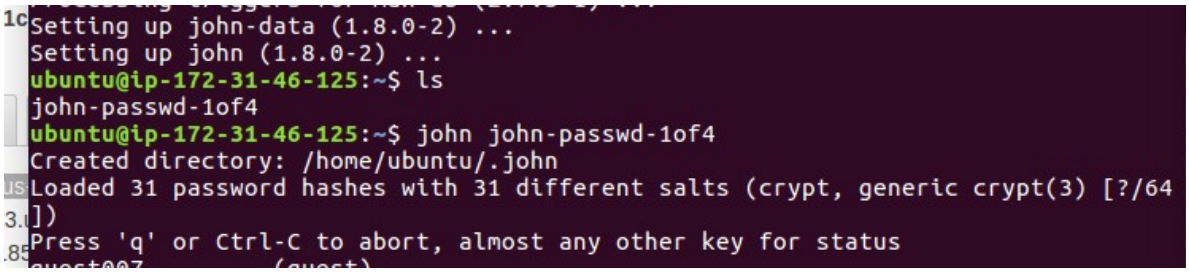


I then logged onto each instance over ssh and installed and ran john on each instance's word list

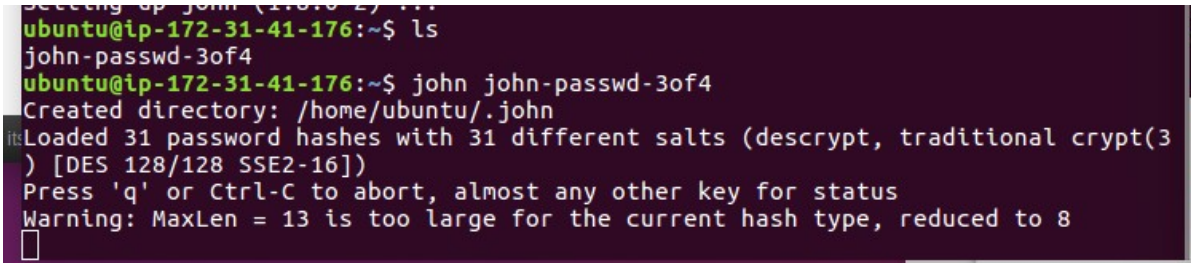
Instance 1



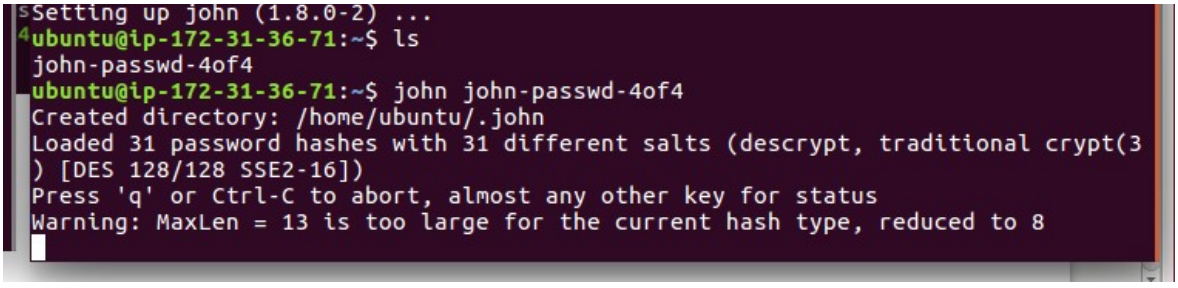
Instance 2



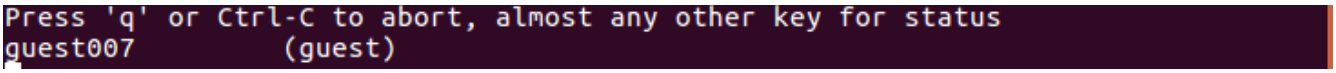
Instance 3



Instance 4



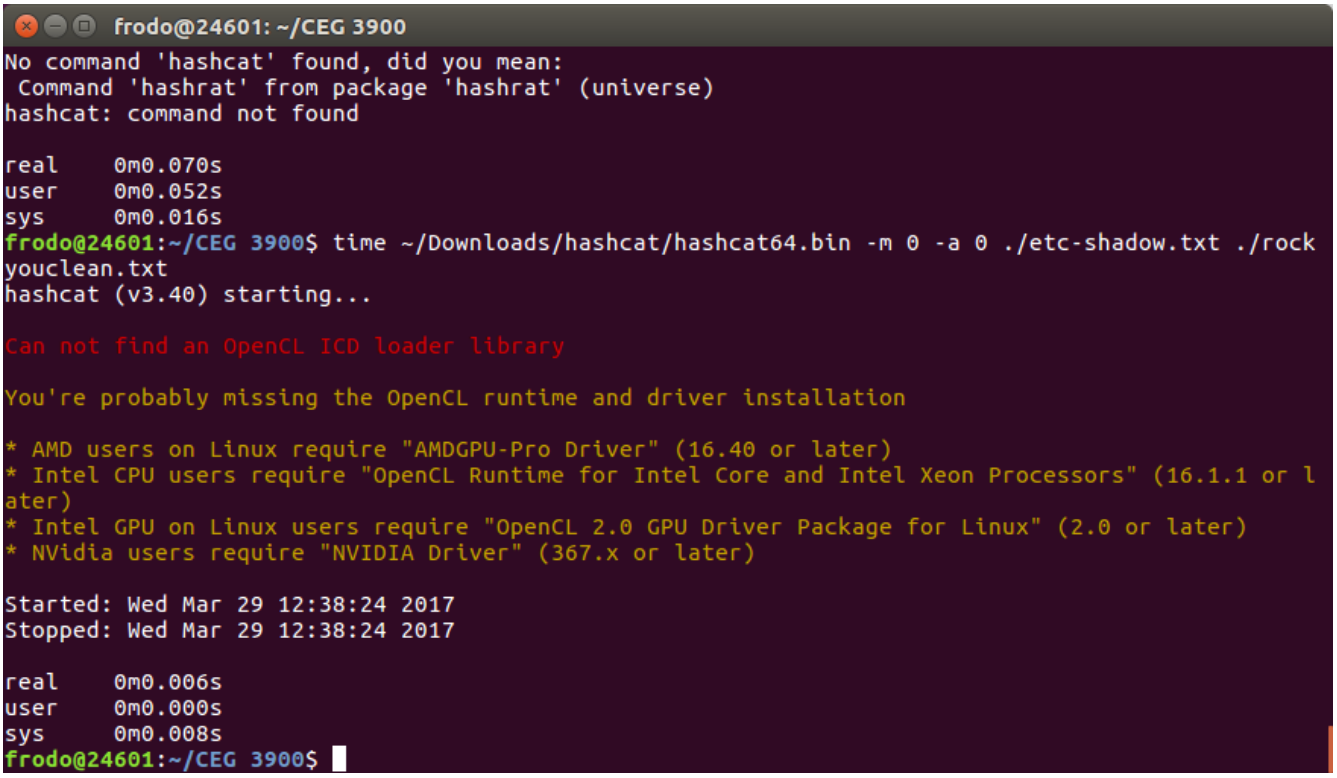
After running for several hours, the only password I recovered was the guest one shown below



3.2 Hashcat (5 hours)

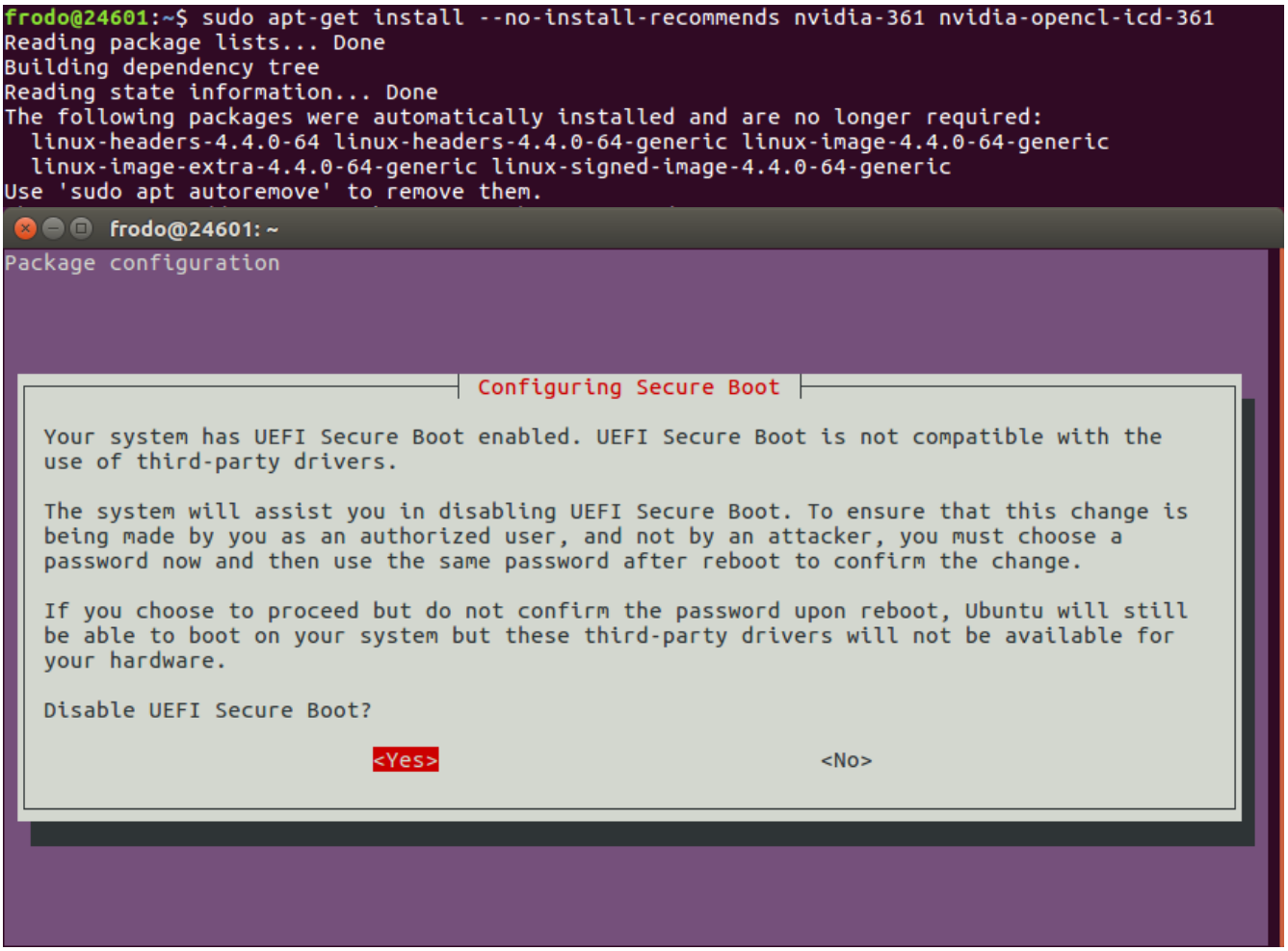
Installation and Drivers

I downloaded hashcat from https://hashcat.net/files/hashcat-3.40.7z. After installing p7zip to unzip the 7z file, I got the below error when trying to run:



At this point, I spent a good deal of time trying to install the packages referenced in the above error message by downloading them from Intel and installing them with rpm or alien. This led to a dead end as the intel packages contained several packages, non

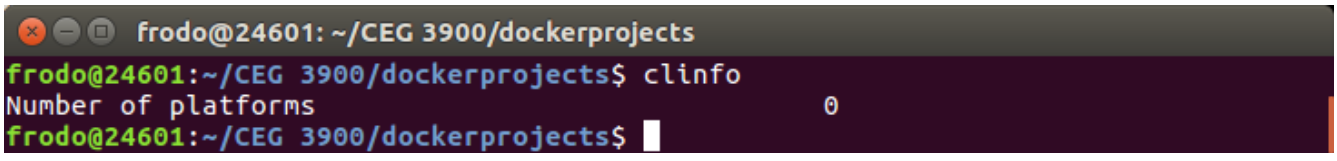
I installed nvidia361, clinfo, and ocl-icd-libopencl1 packages
https://hashcat.net/forum/thread-6063.html

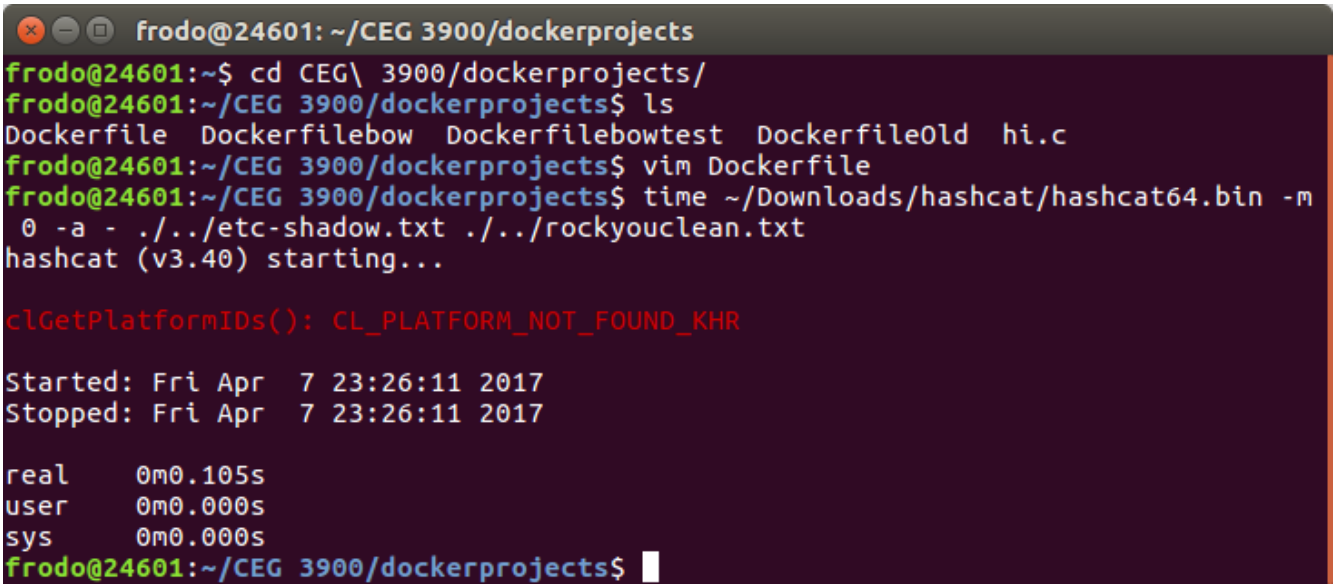


This required disabling secure boot shown above.

I also had to install clinfo and ocl-icd-libopencl1.

When trying to run hashcat, I now get a CL_PLATFORM_NOT_FOUND_KHR error. Running clinfo shows no recognized platforms which I would expect to see my CPU or GPU. This probably means I misinstalled or forgot to install one of the drivers needed to run opencl. For the second part of this project, I will need to determine the cause of this issue.

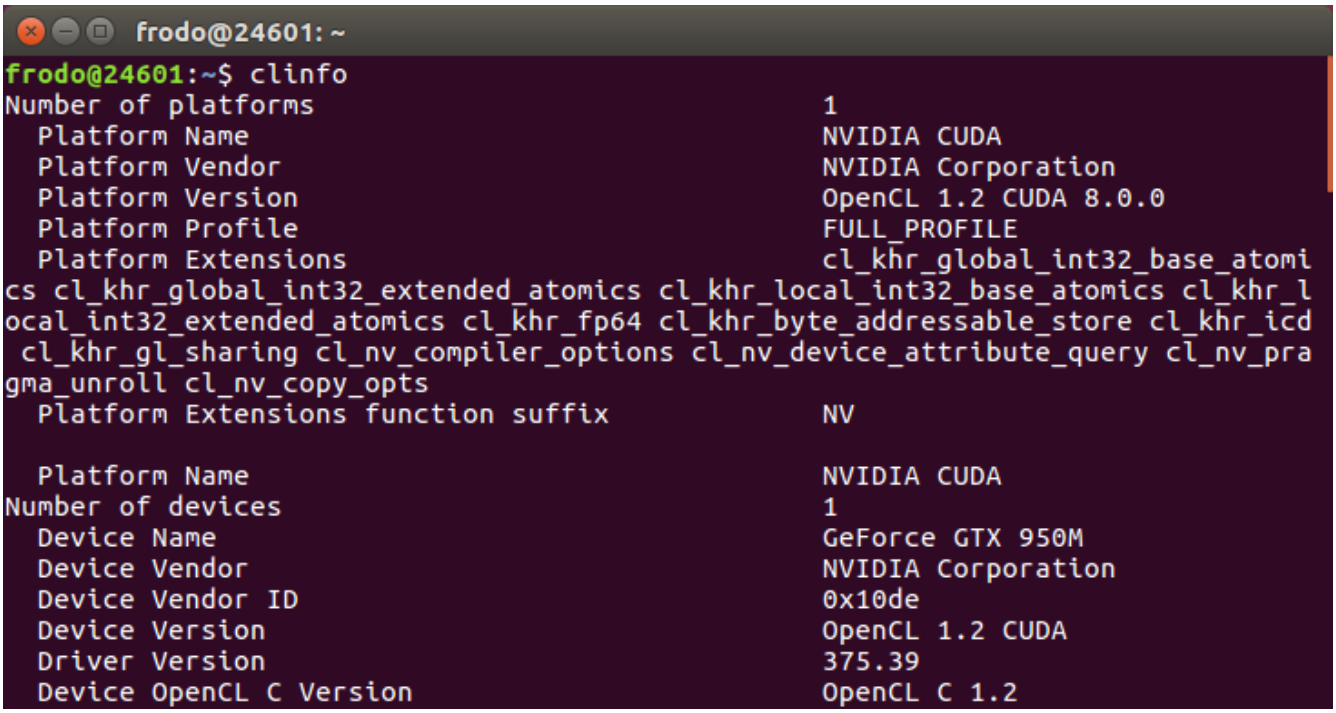




At this point, I decided to try and reinstall the nvidia drivers. In hindsight, I did not properly disable secure boot the first time I installed the nvidia driver.

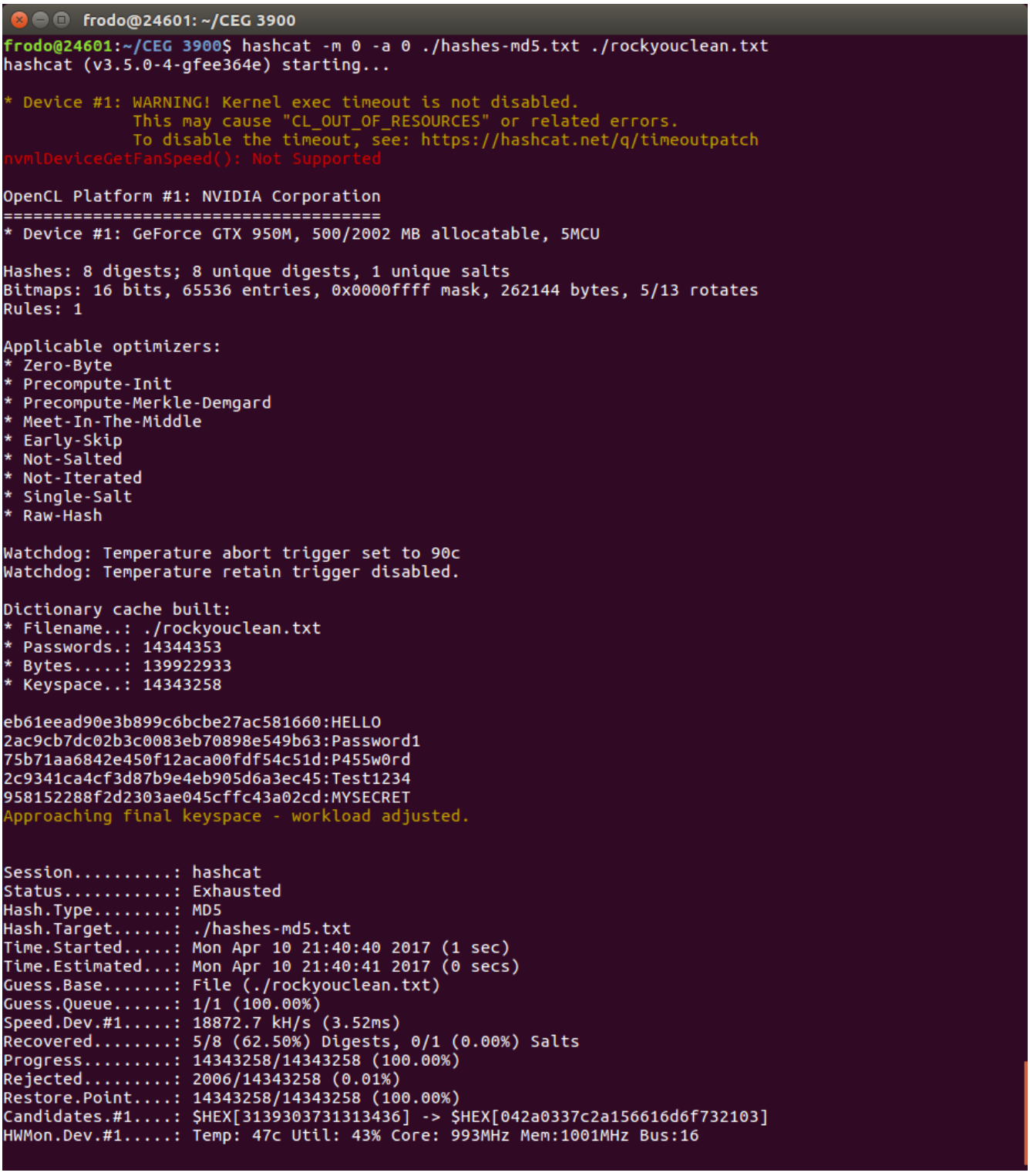
I made the mistake of installing the wrong drivers, installing nividia-default (version 304) instead of nvidia 361 or 375 driver needed by my graphics card (950M). After rebooting and disabling secure boot, this led to an issue where my computer would not boot properly. It gave a “low graphichs mode” error message and then would go to an unresponsive screen. To fix this, I had to disable the x server in grub, login through tty, and reinstall the proper drivers.

Now clinfo showed that I had a platform installed.



I also installed hashcat by cloning it from github and building it with make as outlined in https://github.com/hashcat/hashcat/blob/master/BUILD.md.

And hashcat worked when I ran it!!!



Android APK

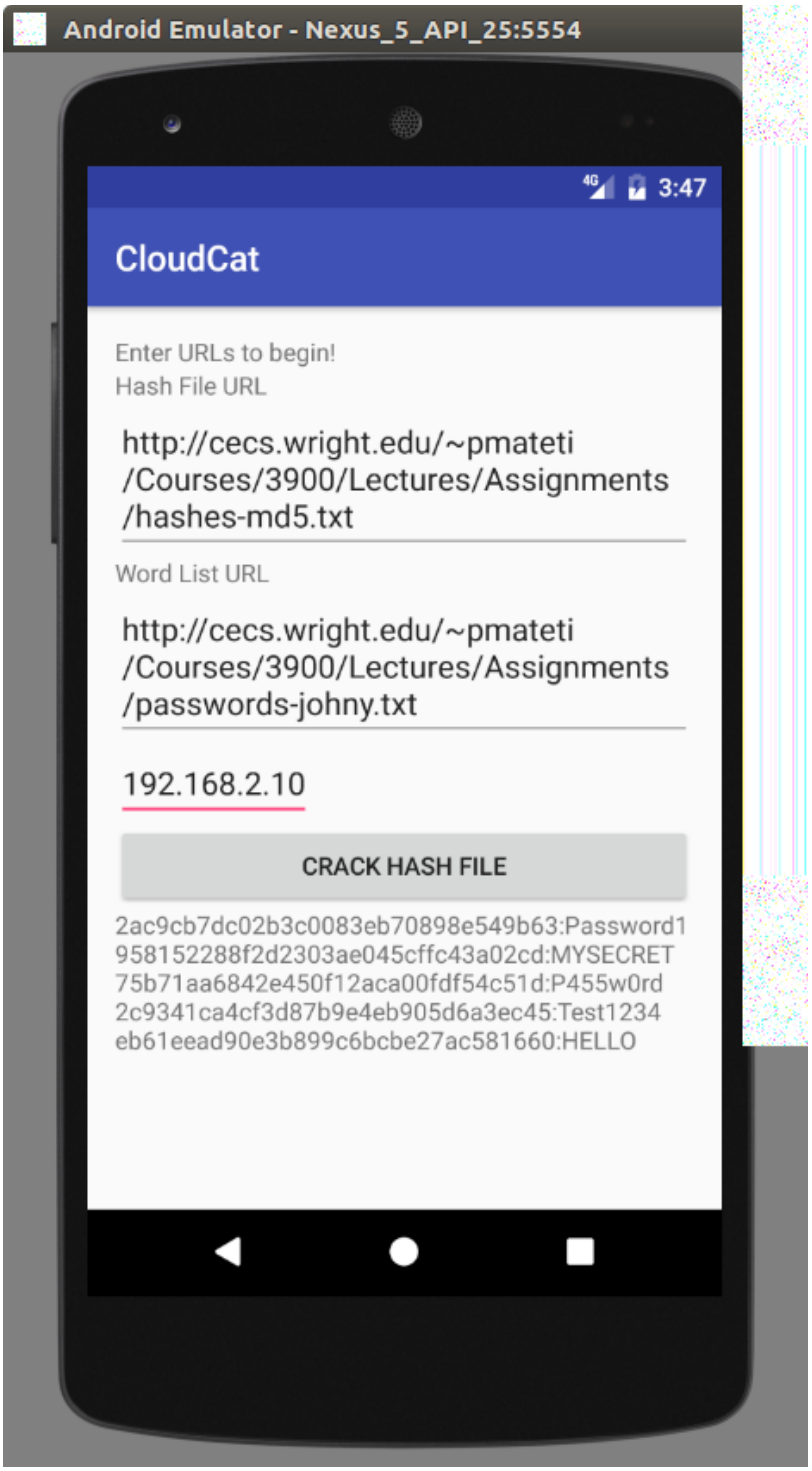
I accomplished this task by running a Java server on my laptop which could be connected to by the android emulator.

The server code is essentially the same as that from P4, reading a string from the client's connection, processing the request, and returning a string back to the client. The only difference is how the client's input is processed. The client passes two arguments seperated by a newline character: the url for the hash file and the url for the word list file. These two inputs are passed to the processRequest method shown below. This method downloads the hash and wordlist files from their urls to hashes.txt and wordlist.txt respectively and then creates a hashcat process that operates on those files. Standard output of the hashcat process is sent back to the client.

```
	public static String processRequest(String hashurl, String wordlisturl)
	{
		String output = "Internal Error";
		try{
			//read files from urls and write to local storage

			//download hash file
			URL hashfileurl = new URL(hashurl);
			String hashes = new Scanner(hashfileurl.openStream(),
				"UTF-8").useDelimiter("\\A").next();
			Files.write(Paths.get("./hashes.txt"), hashes.getBytes());

			//download word list file
			URL wordlistfileurl = new URL(wordlisturl);
			String wordlist = new Scanner(wordlistfileurl.openStream(),
				"UTF-8").useDelimiter("\\A").next();
			Files.write(Paths.get("./wordlist.txt"), wordlist.getBytes());

			//invoke hashcat on the files

			//start the process
			Process hashcat = Runtime.getRuntime().exec("hashcat -a 0 "+
				"--show ./hashes.txt ./wordlist.txt");

			//read the output from hashcat
			Scanner s = new Scanner(hashcat.getInputStream());
			output = "";
			while(s.hasNextLine())
				output += s.nextLine() + "\n";

		}catch (IOException e)
		{e.printStackTrace();}

		return output;
	}

}
```

Screenshot

Below is a picture of the running APK. Note that 192.168.2.10 was the address of my laptop running the server.

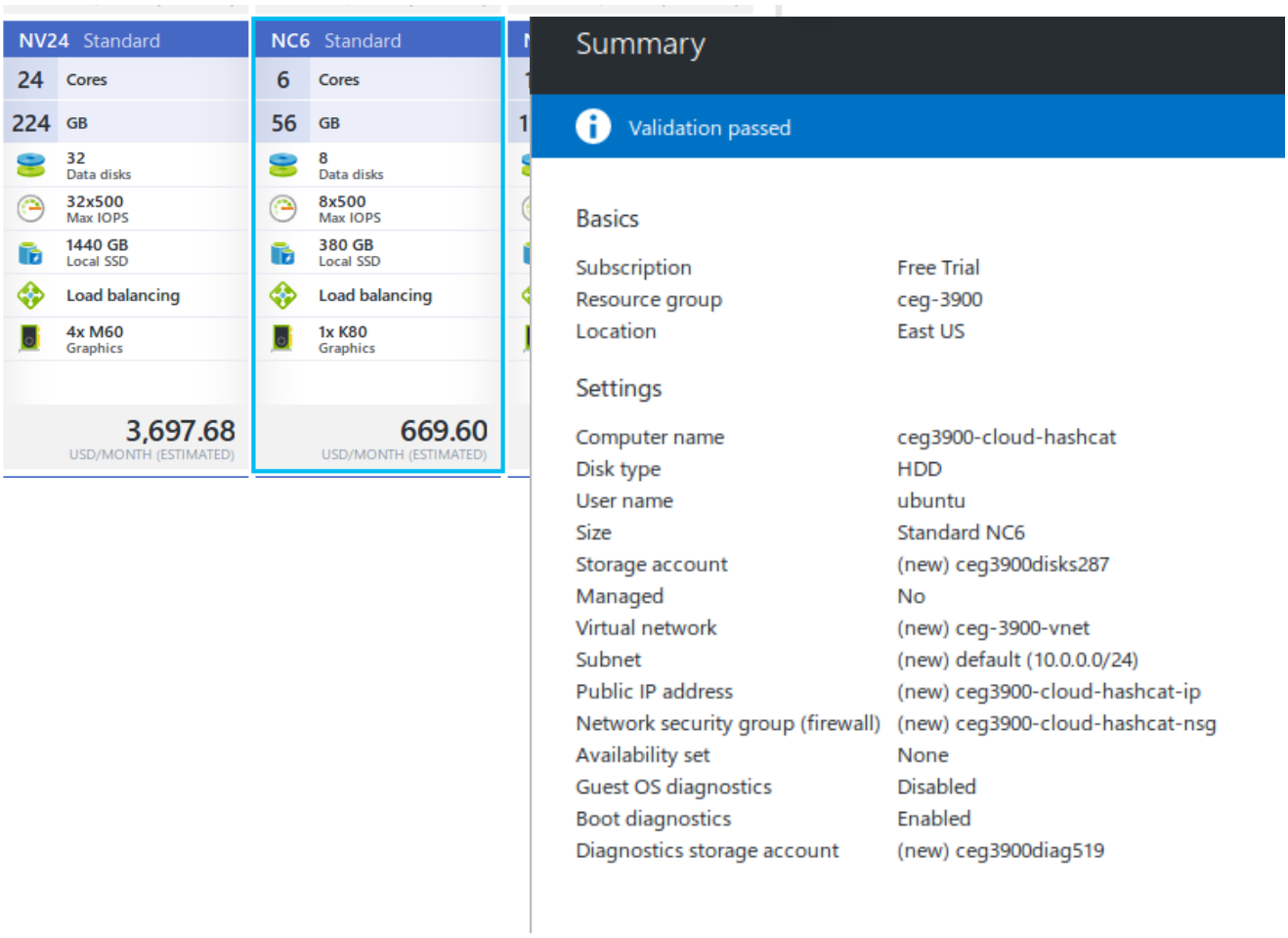


3.3 Hashcat in the Cloud (3 hours)

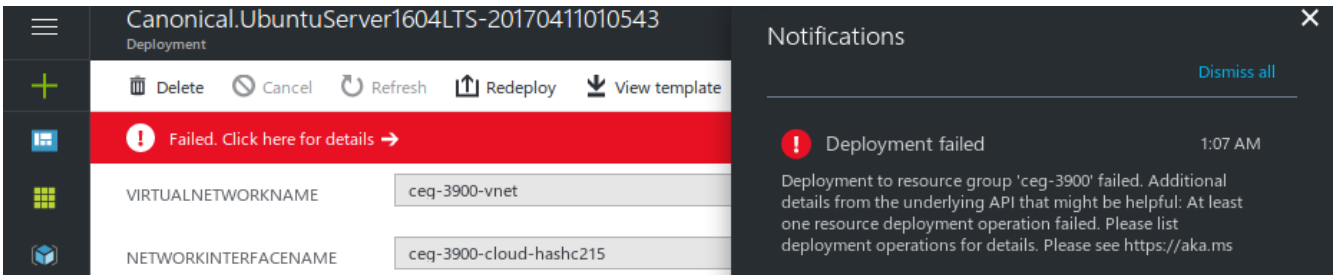
Azure Attempt

First, I found an article on how to use hashcat with Azure, so I decided to try it out.
http://javydekoning.com/recover-crack-password-hashes-using-azure-gpu-powered-n-series-virtual-machines-and-hashcat/

Choosing a VM
I went with NC6, the smallest VM with a GPU to run hashcat on.



However, this approach proved to not work as Azure's free tier only allows VMs with up to four cores.

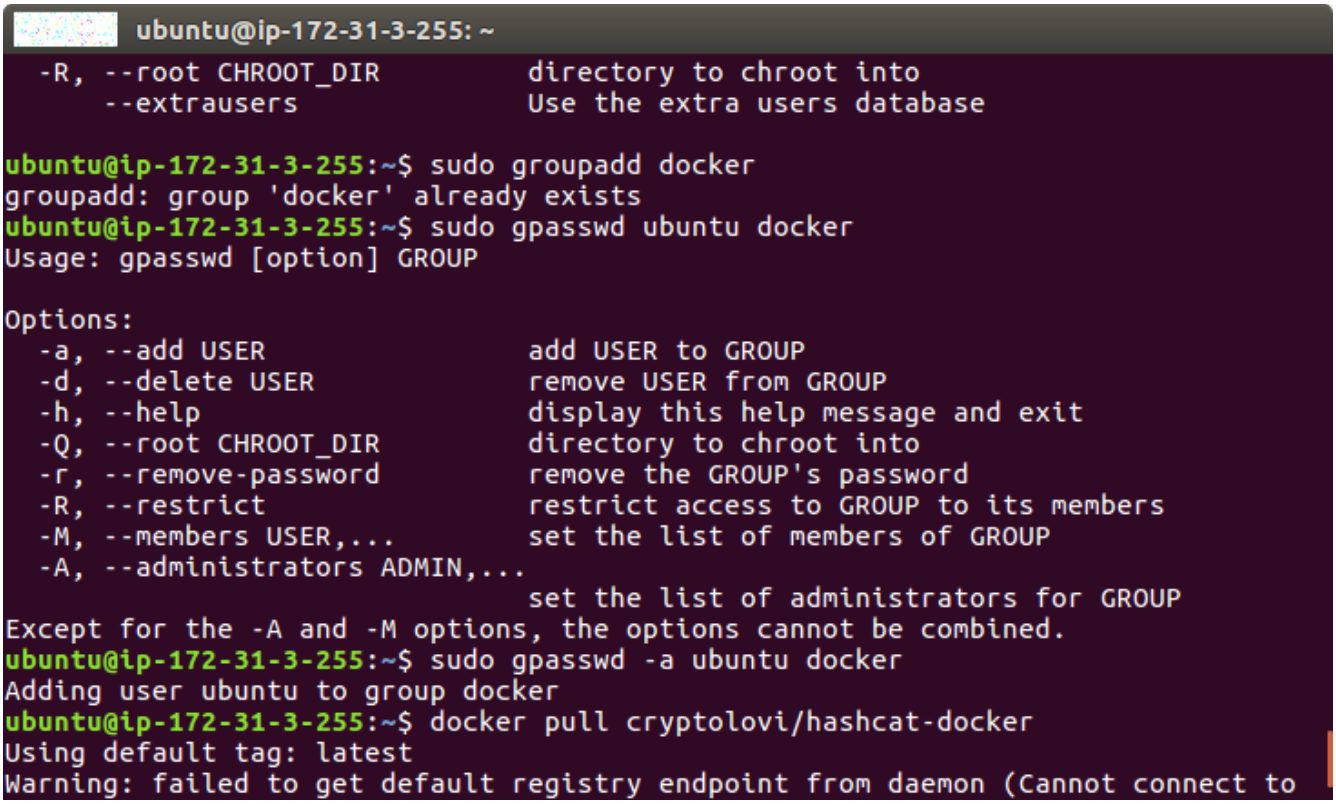


AWS and Docker

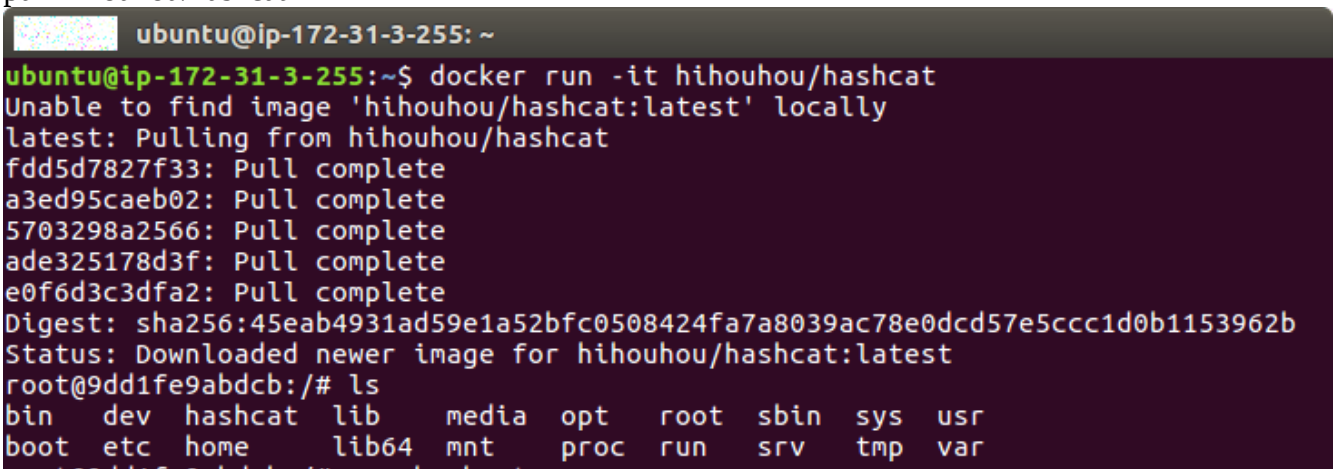
Next, I switched to AWS to finish the task.

To avoid the troubles I had installing hashcat on my own laptop, I decided to try and run hashcat from an already existing docker image. After some experimentation, hihouhou/hashcat was the only one I found that seemed to work as is.

Install docker.io



pull hihouhou/hashcat



Next, I used wget inside the docker to download the hashes-md5.txt from the P6 webpage and rockyou from github/SecLists. Running hashcat with these two files cracked two of the passwords (on my laptop I got five).

```
ubuntu@ip-172-31-3-255: ~
root@9dd1fe9abdcb:~# cd /hash
bash: cd: /hash: No such file or directory
root@9dd1fe9abdcb:~# cd /hashcat
root@9dd1fe9abdcb:/hashcat# hashcat  -m 0 -a 0 hashes-md5.txt rockyou-75.txt
Initializing hashcat v2.00 with 1 threads and 32mb segment-size...

Added hashes from file hashes-md5.txt: 8 (1 salts)

2ac9cb7dc02b3c0083eb70898e549b63:Password1
eb61eead90e3b899c6bcbe27ac581660:HELLO
```

By experimentation, I discovered that if I used a rules file (T0XlC.rule), I could get five passswords

```
ubuntu@ip-172-31-3-255: ~
root@9dd1fe9abdcb:/hashcat# hashcat  -m 0 -a 0 -r T0XlC.rule hashes-md5.txt rock
you-75.txt
Initializing hashcat v2.00 with 1 threads and 32mb segment-size...

Added hashes from file hashes-md5.txt: 8 (1 salts)
Added rules from file T0XlC.rule: 4086

2ac9cb7dc02b3c0083eb70898e549b63:Password1
eb61eead90e3b899c6bcbe27ac581660:HELLO
75b71aa6842e450f12aca00fdf54c51d:P455w0rd
958152288f2d2303ae045cffc43a02cd:MYSECRET
2c9341ca4cf3d87b9e4eb905d6a3ec45:Test1234
```

I spent a while here figuring out how to run hashcat on the docker image to verify that it worked
I also spent time trying to pass the files into the container via command line when running the container so my Java server could run on aws and invoke the docker container to run hashcat. After some unsecessful attemps, I realized that I could run the Java server in the docker container and use port forewording to make it visible to the outside world

First I ran the hihouhou/hashcat docker image forewarding port 8080

```
ubuntu@ip-172-31-3-255:~$ docker run -it -p8080:8080 hihouhou/hashcat
root@07a521bab40f:/# ls
CloudCatServer.class  dev      home   media  proc  sbin  tmp
bin                   etc      lib    mnt    root  srv   usr
boot                  hashcat  lib64  opt    run   sys   var
root@07a521bab40f:/#
```

Then I copied my java server from 3.2 to the AWS machine via scp.
And copied my java server from AWS machine to running docker container using docker cp (from a separate ssh session)

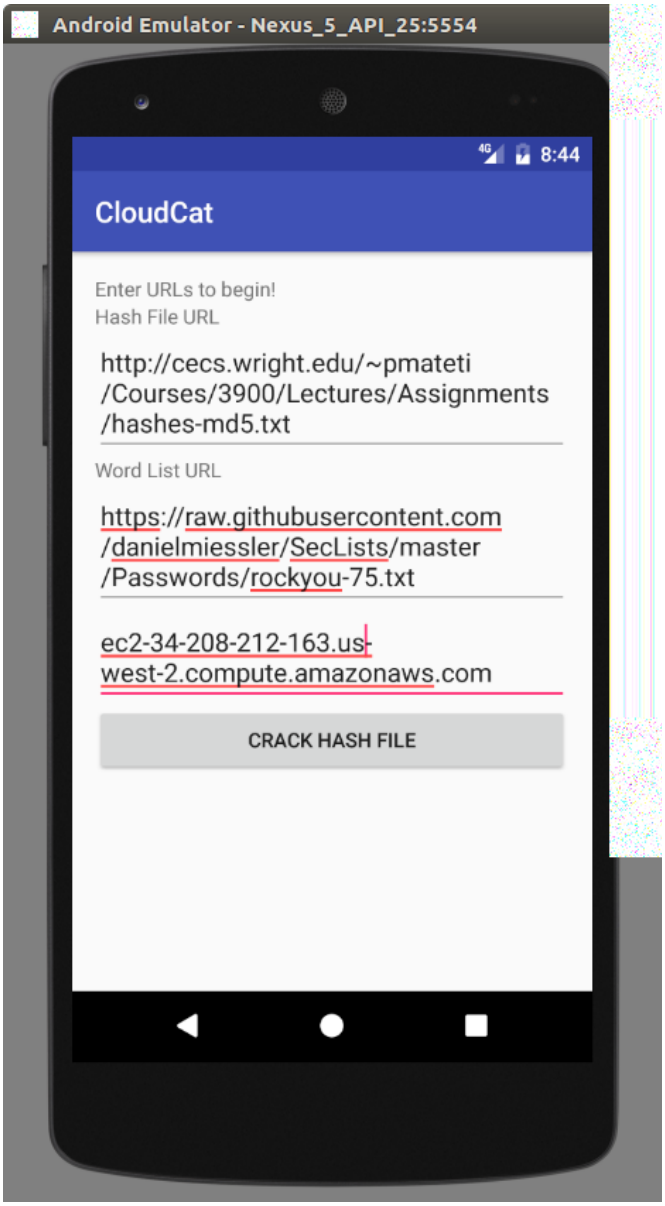
```
ubuntu@ip-172-31-3-255:~$ docker cp CloudCatServer.class 07a521bab40f2f7e510a952
317e43d1facd8e159dc5b343a49797647d24ab33d:/CloudCatServer.class
ubuntu@ip-172-31-3-255:~$
```

I then installed a jdk on in the aws container. Since the container is kind of old, I could only install Java 7. Because of this, I had to modify my server code slightly to make it Java 7 compatible (replace a lambda with an anonymous type).

After that, I compiled and ran the Java CloudCat server inside the docker

```
ubuntu@ip-172-31-3-255: ~
^[[Aroot@07a521bab40f:/# javac CloudCatServer.java
root@07a521bab40f:/# java CloudCatServer
Started server on port 8080
Accpted Connection
```

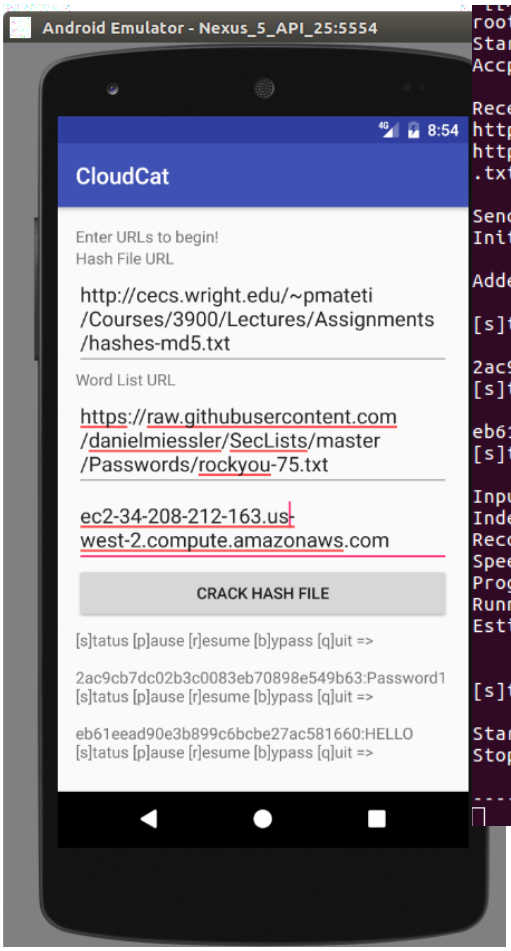
Finally, I ran the android APK from 3.3 using my AWS instance's host name and port 8080 (which is foreworded to the hashcat docker container running the Java server)



Screenshots

The android APK sent the urls to the server running inside docker on AWS which ran hashcat and responded with hashcat's output. This output is displayed in a scroll view at the bottom of the andorid activity.

Client



Server

```
root@07a521bab40f:/# java CloudCatServer
Started server on port 8080
Accpted Connection

Received Request:
http://cecs.wright.edu/~pmateti/Courses/3900/Lectures/Assignments/hashes-md5.txt
https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/rockyou-75
.txt

Sending Response:
Initializing hashcat v2.00 with 1 threads and 32mb segment-size...

Added hashes from file ./hashes.txt: 8 (1 salts)

[s]tatus [p]ause [r]esume [b]ypass [q]uit =>

2ac9cb7dc02b3c0083eb70898e549b63:Password1
[s]tatus [p]ause [r]esume [b]ypass [q]uit =>

eb61eead90e3b899c6bcbe27ac581660:HELLO
[s]tatus [p]ause [r]esume [b]ypass [q]uit =>

Input.Mode: Dict (./wordlist.txt)
Index.....: 1/1 (segment), 59187 (words), 478925 (bytes)
Recovered.: 2/8 hashes, 0/1 salts
Speed/sec.: - plains, - words
Progress..: 59187/59187 (100.00%)
Running...: --:--:--:--
Estimated.: --:--:--:--


[s]tatus [p]ause [r]esume [b]ypass [q]uit =>

Started: Thu Apr 13 03:48:51 2017
Stopped: Thu Apr 13 03:48:52 2017

------------------------------------------------------------
```

3.4 Docker (6 hours)

Installation

Initially, I could not connect to deamon since I had not done the group groupadd and gpasswd

```
frodo@24601:~$ docker run --rm -ti ubuntu /bin/bash
docker: Cannot connect to the Docker daemon. Is the docker daemon running on thi
s host?.
See 'docker run --help'.
frodo@24601:~$ sudo groupadd docker
groupadd: group 'docker' already exists
frodo@24601:~$ sudo gpasswd -a frodo docker
Adding user frodo to group docker
frodo@24601:~$
```

After logging out and back in and executing ‘sudo service docker restart’

Docker ps worked!

```
frodo@24601: ~
frodo@24601:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
frodo@24601:~$
```

Running a bash on an ubuntu image to verify docker is working

```
root@8e3d6f102ad2: /
frodo@24601:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
frodo@24601:~$ docker run --rm -ti ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu

d54efb8db41d: Pull complete
f8b845f45a87: Pull complete
e8db7bf7c39f: Pull complete
9654c40e9079: Pull complete
6d9ef359eaaa: Pull complete
Digest: sha256:dd7808d8792c9841d0b460122f1acf0a2dd1f56404f8d1e56298048885e45535
Status: Downloaded newer image for ubuntu:latest
root@8e3d6f102ad2:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@8e3d6f102ad2:/# echo $SHELL
/bin/bash
root@8e3d6f102ad2:/#
```

Creating a docker image

I began by building some simple docker files to get a feel for how it worked before tackling painbow. To try it on a simple project, I used a short program hi.c. hi.c prints the size of various primitive types.

```
frodo@24601:~/CEG 3900/dockerprojects$ ./hi
Size of int: 4
Size of char: 1
Size of ling: 8
frodo@24601:~/CEG 3900/dockerprojects$
```

Creating the Dockerfile
The docker file compiles hi.c on an ubuntu 16.04 image and then runs the resulting executable

```
frodo@24601: ~/CEG 3900/dockerprojects
FROM ubuntu:16.04

RUN apt-get -y update && apt-get install -y clang

CMD clang hi.c -o hi; ./hi

                                                              1,1           All
```

Building the docker file
I used the command ‘docker build -t hi.c .’ to construct the container.

Docker images command shows the containers docker can run.

```
frodo@24601:~/CEG 3900/dockerprojects$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
hi.c                latest              bbcb735dc465        4 minutes ago
564.2 MB
ubuntu              16.04               0ef2e08ed3fa        4 weeks ago
130 MB
ubuntu              latest              0ef2e08ed3fa        4 weeks ago
130 MB
```

Running my container

```
frodo@24601:~/CEG 3900/dockerprojects$ docker run hi.c
clang: error: no such file or directory: 'hi.c'
clang: error: no input files
/bin/sh: 1: ./hi: not found
frodo@24601:~/CEG 3900/dockerprojects$
```

Whoops! It appears my file hi.c did not get included in the image. I would have to add a line under RUN which copies the hi.c file into the container. Howerver, since I accomplished by goal of proving I could use docker, I moved on the painbow.

Building painbow on my machine

Plan: find commands necessary to install cassandra and painbow on Ubuntu
Use the same commands in the docker file using an ubuntu 16.04 image to build the container

Install Necessary Software

According to the GitHub page, I need the following installed
Cassandra 2+
JDK 1.7+
gradle 2.1+

:Cassandra
Instructions: http://cassandra.apache.org/download/

```
frodo@24601:~/CEG 3900$ echo "deb http://www.apache.org/dist/cassandra/debian 31
0x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
deb http://www.apache.org/dist/cassandra/debian 310x main
frodo@24601:~/CEG 3900$ curl https://www.apache.org/dist/cassandra/KEYS | sudo a
pt-key add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  244k  100  244k    0     0   171k      0  0:00:01  0:00:01 --:--:--  171k
OK
frodo@24601:~/CEG 3900$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Hit:4 http://archive.canonical.com/ubuntu xenial InRelease
Get:6 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [235
 kB]
frodo@24601:~/CEG 3900$ sudo apt-get install cassandra
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libopts25 ntp
Suggested packages:
  cassandra-tools ntp-doc
The following NEW packages will be installed:
  cassandra libopts25 ntp
0 upgraded, 3 newly installed, 0 to remove and 246 not upgraded.
Need to get 29.7 MB of archives.
After this operation, 40.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Gradle 2.1 wase installed via ‘apt install gradle’
JDK 8 was installed with ‘apt install default-jdk’

Building Painbow Locally

Install git to download the repo ‘sudo apt-get install git’

sudo service cassandra start

git clone git:https://github.com/mcandre/painbow.git
cd painbow
gradle build
cd build/classes/main

java us.yellosoft.painbow.Painbow
ERROR!! dependancies not included

cd build/libs/

Resolving dependencies

I manually downloaded the JAR file dependencies specified in painbow’s build file from the maven central repository.

wget http://central.maven.org/maven2/com/offbytwo/docopt/0.6.0.20150202/docopt-0.6.0.20150202.jar
/usr/share/cassandra/lib/*
wget http://central.maven.org/maven2/commons-codec/commons-codec/1.10/commons-codec-1.10.jar

Finally, painbow was run using the below command.

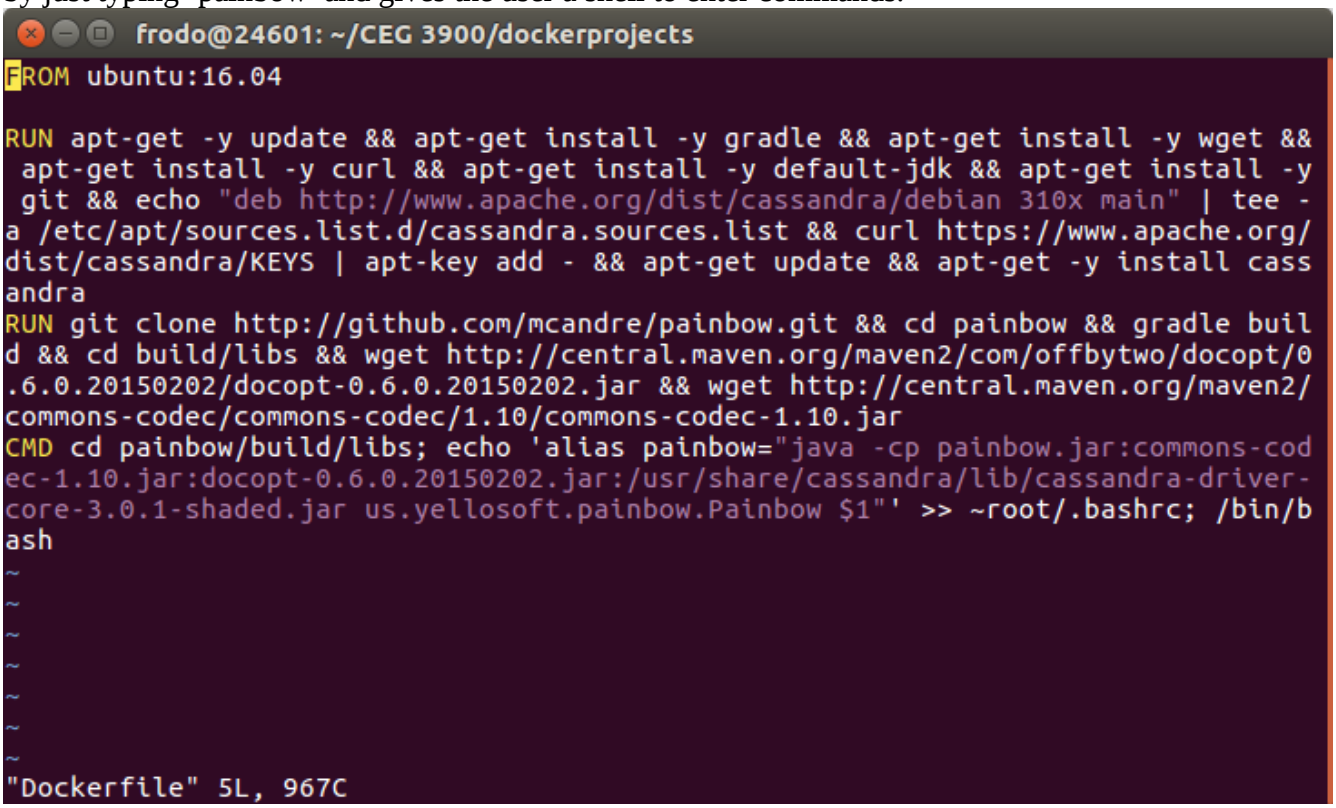
java -cp 'painbow.jar:commons-codec-1.10.jar:docopt-0.6.0.20150202.jar:/usr/share/cassandra/lib/cassandra-driver-core-3.0.1-shaded.jar' us.yellosoft.painbow.Painbow

On the next page is a screen shot showing these commands.

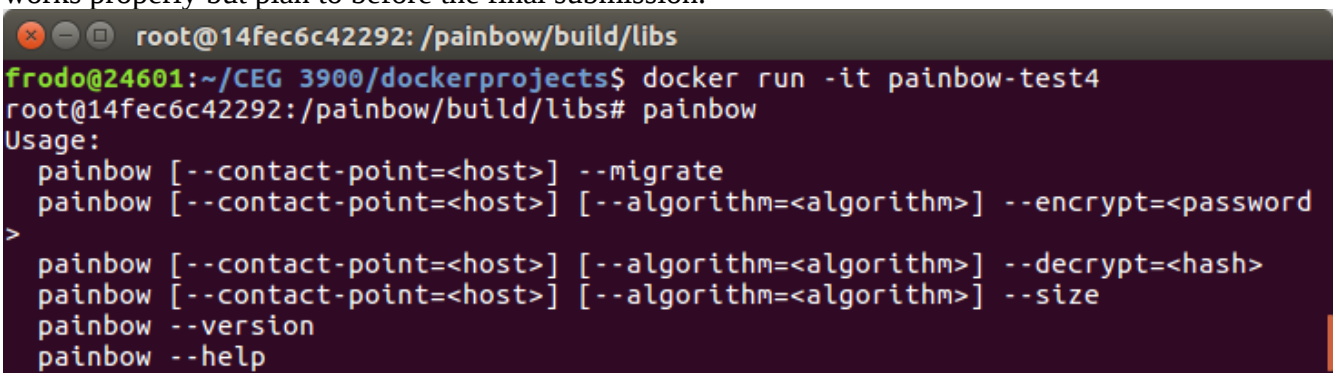
```
frodo@24601:~/CEG 3900/painbow-test/painbow/build/libs$ wget http://central.mave
n.org/maven2/com/offbytwo/docopt/0.6.0.20150202/docopt-0.6.0.20150202.jar
--2017-04-03 01:32:32--  http://central.maven.org/maven2/com/offbytwo/docopt/0.6
.0.20150202/docopt-0.6.0.20150202.jar
Resolving central.maven.org (central.maven.org)... 151.101.44.209
Connecting to central.maven.org (central.maven.org)|151.101.44.209|:80... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 35943 (35K) [application/java-archive]
Saving to: ‘docopt-0.6.0.20150202.jar’

docopt-0.6.0.201502 100%[===================>]  35.10K  --.-KB/s    in 0.09s

2017-04-03 01:32:32 (398 KB/s) - ‘docopt-0.6.0.20150202.jar’ saved [35943/35943]

frodo@24601:~/CEG 3900/painbow-test/painbow/build/libs$ wget http://central.mave
n.org/maven2/commons-codec/commons-codec/1.10/commons-codec-1.10.jar
--2017-04-03 01:32:42--  http://central.maven.org/maven2/commons-codec/commons-c
odec/1.10/commons-codec-1.10.jar
Resolving central.maven.org (central.maven.org)... 151.101.44.209
Connecting to central.maven.org (central.maven.org)|151.101.44.209|:80... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 284184 (278K) [application/java-archive]
Saving to: ‘commons-codec-1.10.jar’

commons-codec-1.10. 100%[===================>] 277.52K   920KB/s    in 0.3s

2017-04-03 01:32:43 (920 KB/s) - ‘commons-codec-1.10.jar’ saved [284184/284184]

frodo@24601:~/CEG 3900/painbow-test/painbow/build/libs$ java -cp 'painbow.jar:co
mmons-codec-1.10.jar:docopt-0.6.0.20150202.jar:/usr/share/cassandra/lib/cassandr
a-driver-core-3.0.1-shaded.jar' us.yellosoft.painbow.Painbow
Usage:
  painbow [--contact-point=<host>] --migrate
  painbow [--contact-point=<host>] [--algorithm=<algorithm>] --encrypt=<password
>
  painbow [--contact-point=<host>] [--algorithm=<algorithm>] --decrypt=<hash>
  painbow [--contact-point=<host>] [--algorithm=<algorithm>] --size
  painbow --version
  painbow --help
frodo@24601:~/CEG 3900/painbow-test/painbow/build/libs$
```

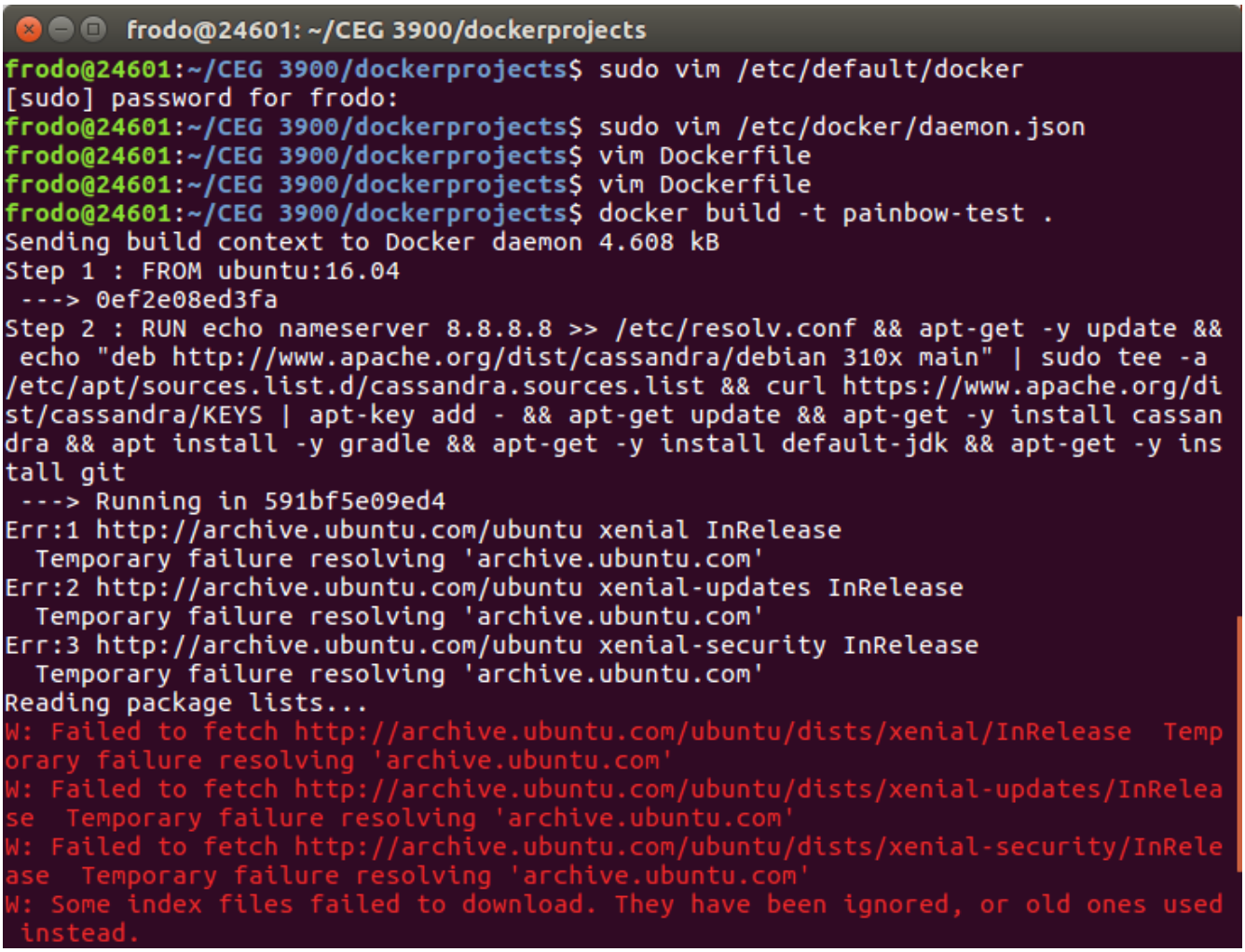
Docker painbow container build

I took the commands necessary to build docker on my laptop and placed them in a dockerfile. The dockerfile RUN downloads all the project's dependencies and builds the painbow project from github. The CMD uses cd to the directory containing painbow, defines an alias so the user can execute painbow by just typing 'painbow' and gives the user a shell to enter commands.



Below shows the command to run my docker container followed by the user running painbow inside the container's shell. I have yet to test painbow with actual commands to ensure the cassandra database works properly but plan to before the final submission.

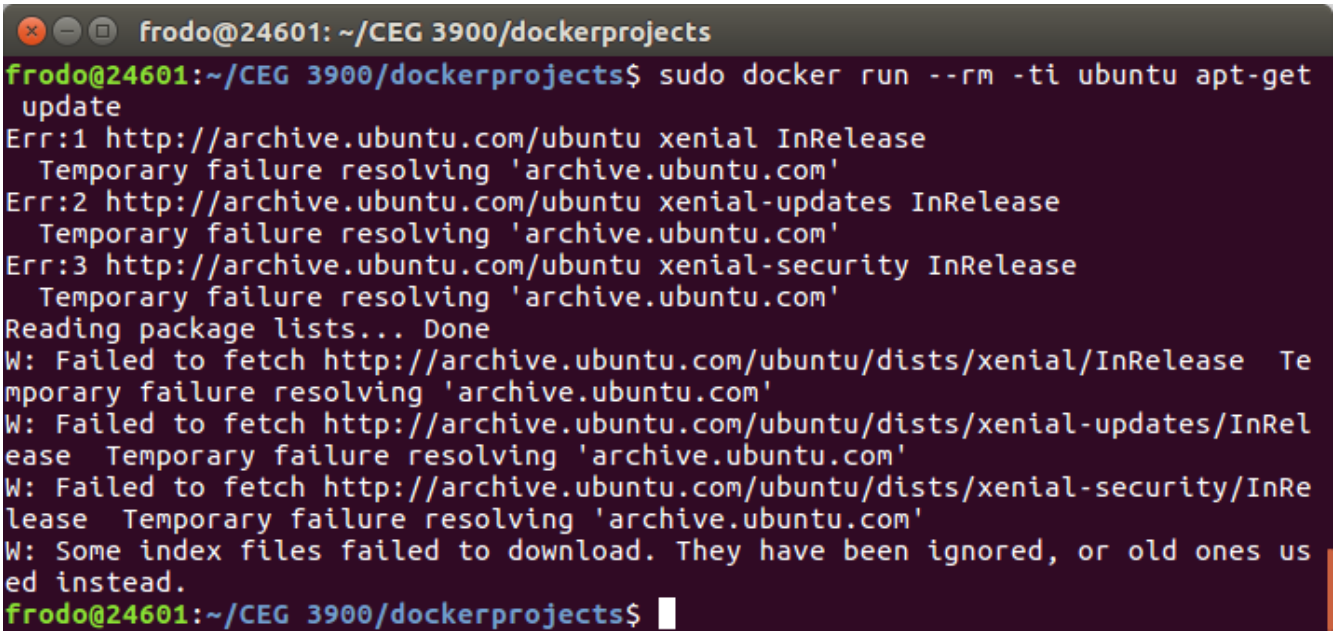




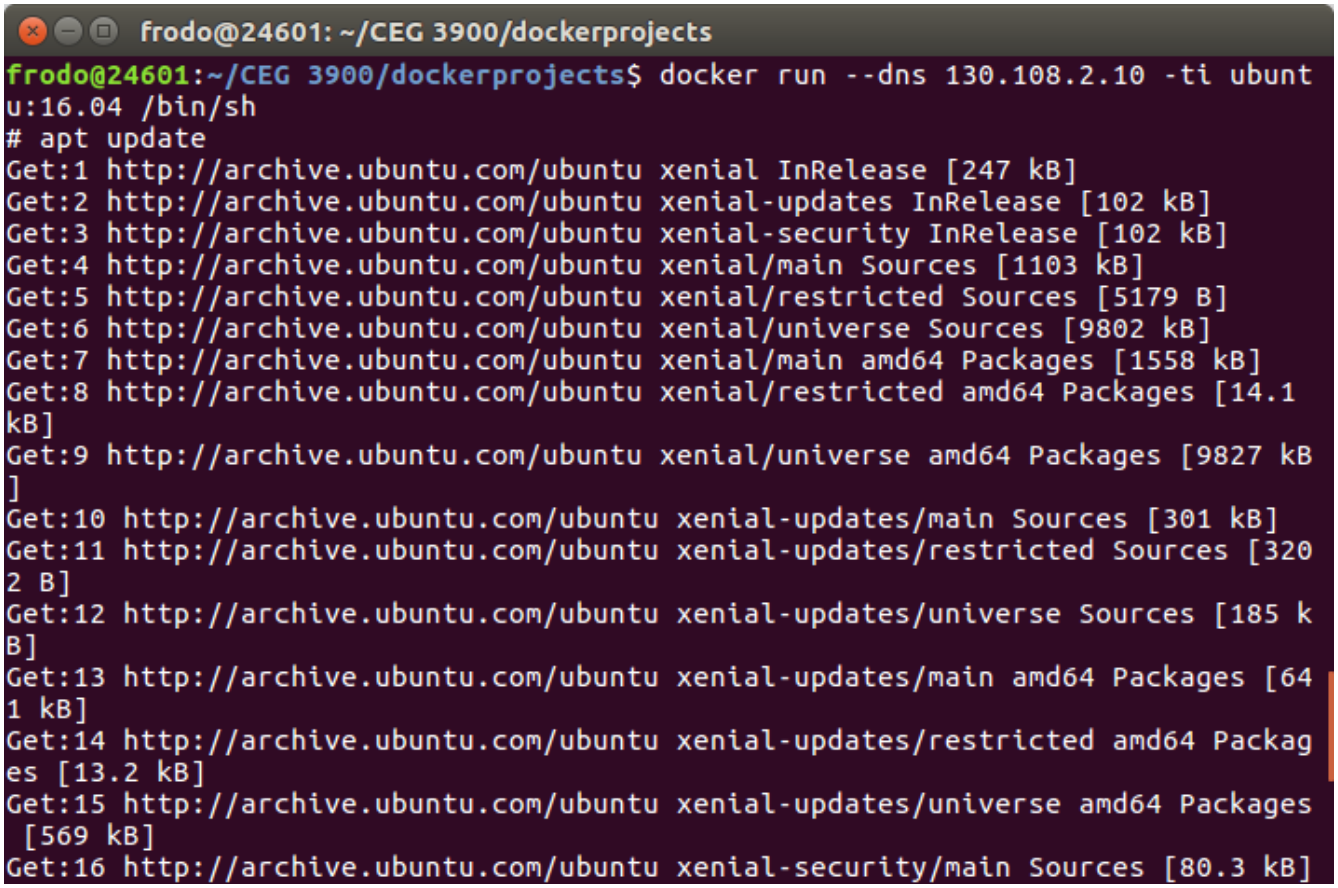
DOCKER_OPTS="--dns 8.8.8.8 --dns 130.108.2.10"

Also this only works on the WSU Student (unsecured) network

Docker can not resolve host ‘archive.ubuntu.com’



After doing a good deal of research, it appears this is a problem with trying to access public DNS servers from behind the WSU firewall. This seems to be supported by the fact that my successful build was performed on a private network.
Adding the WSU dns server to the docker build command solved the issue



For the docker build, I added the line ‘DOCKER_OPTS="--dns 130.108.2.10 --dns 8.8.8.8”’ to /etc/default/docker. 130.108.2.10 is wright state’s dns server.

```
# If that does not suit your needs, try a systemd drop-in file, as described in:
#   https://docs.docker.com/v1.11/engine/admin/systemd/#custom-docker-daemon-opt
ions

DOCKER_OPTS="--dns 8.8.8.8 --dns 8.8.4.4"
~
```

Below shows that docker now builds successfully

```
frodo@24601:~/CEG 3900/dockerprojects$ sudo vim /etc/default/docker
frodo@24601:~/CEG 3900/dockerprojects$ sudo service docker start
frodo@24601:~/CEG 3900/dockerprojects$ sudo service docker restart
frodo@24601:~/CEG 3900/dockerprojects$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
  STATUS              PORTS               NAMES
frodo@24601:~/CEG 3900/dockerprojects$ sudo docker build -t test .
Sending build context to Docker daemon 5.632 kB
Step 1 : FROM ubuntu:16.04
 ---> 0ef2e08ed3fa
Step 2 : RUN apt-get -y update && apt-get install -y vim
 ---> Running in 264d6c894922
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial/main Sources [1103 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial/restricted Sources [5179 B]
Get:6 http://archive.ubuntu.com/ubuntu xenial/universe Sources [9802 kB]
```

After a bit of debugging, I arrived at the sequence of docker commands. When run, the container defines an alias ‘painbow’ to run the painbow jar and gives the user a shell as shown at the beginning of the section.

6b Update on Painbow

Running a command triggers a class not found error

```
frodo@24601:~/CEG 3900/painbow-test/painbow/build/libs$ java -cp 'painbow.jar:co
mmons-codec-1.10.jar:docopt-0.6.0.20150202.jar:/usr/share/cassandra/lib/cassandr
a-driver-core-3.0.1-shaded.jar' us.yellosoft.painbow.Painbow -e "hi"
Exception in thread "main" java.lang.NoClassDefFoundError: com/google/common/uti
l/concurrent/AsyncFunction
        at us.yellosoft.painbow.Painbow.main(Painbow.java:153)
Caused by: java.lang.ClassNotFoundException: com.google.common.util.concurrent.A
syncFunction
        at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
        ... 1 more
frodo@24601:~/CEG 3900/painbow-test/painbow/build/libs$
```

Solution:
Change command to include all jars in cassandra using *
Also, I added a call to ‘service cassandra start’ to the CMD

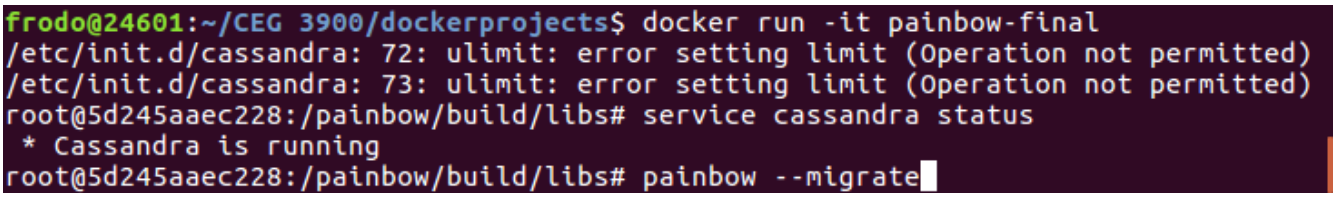
java -cp 'painbow.jar:commons-codec-1.10.jar:docopt-0.6.0.20150202.jar:/usr/share/cassandra/lib/*' us.yellosoft.painbow.Painbow

Final Docker file

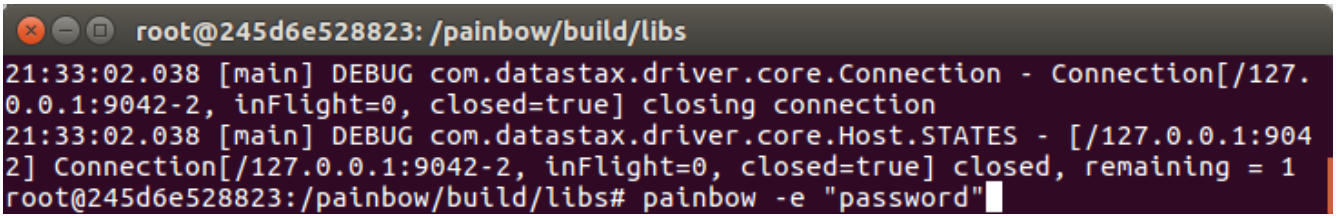
```
frodo@24601: ~/CEG 3900/dockerprojects
FROM ubuntu:16.04

RUN apt-get -y update && apt-get install -y gradle && apt-get install -y wget && a
pt-get install -y curl && apt-get install -y default-jdk && apt-get install -y git
 && echo "deb http://www.apache.org/dist/cassandra/debian 310x main" | tee -a /etc
/apt/sources.list.d/cassandra.sources.list && curl https://www.apache.org/dist/cas
sandra/KEYS | apt-key add - && apt-get update && apt-get -y install cassandra
RUN git clone http://github.com/mcandre/painbow.git && cd painbow && gradle build
&& cd build/libs && wget http://central.maven.org/maven2/com/offbytwo/docopt/0.6.0
.20150202/docopt-0.6.0.20150202.jar && wget http://central.maven.org/maven2/common
s-codec/commons-codec/1.10/commons-codec-1.10.jar
CMD cd painbow/build/libs; echo 'alias painbow="java -cp painbow.jar:commons-codec
-1.10.jar:docopt-0.6.0.20150202.jar:/usr/share/cassandra/lib/* us.yellosoft.painbo
w.Painbow $1"' >> ~root/.bashrc; service cassandra start; /bin/bash
~
                                                              2,0-1         All
```

Final Screenshots

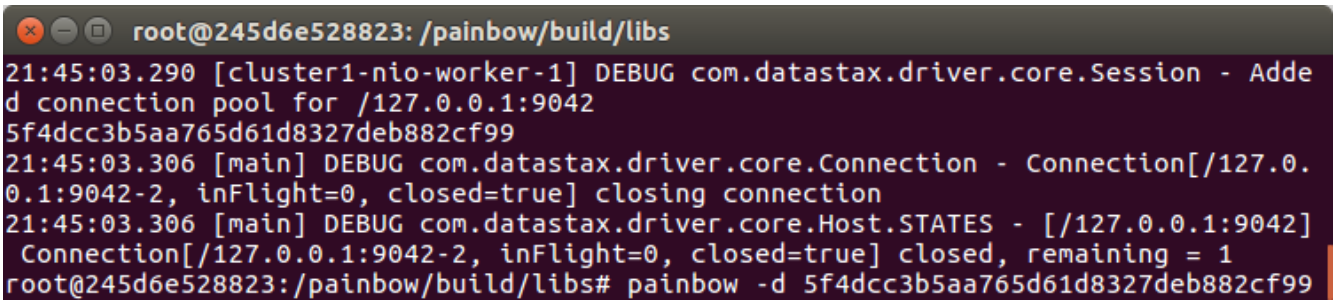
Start the docker and call painbow migrate to configure it



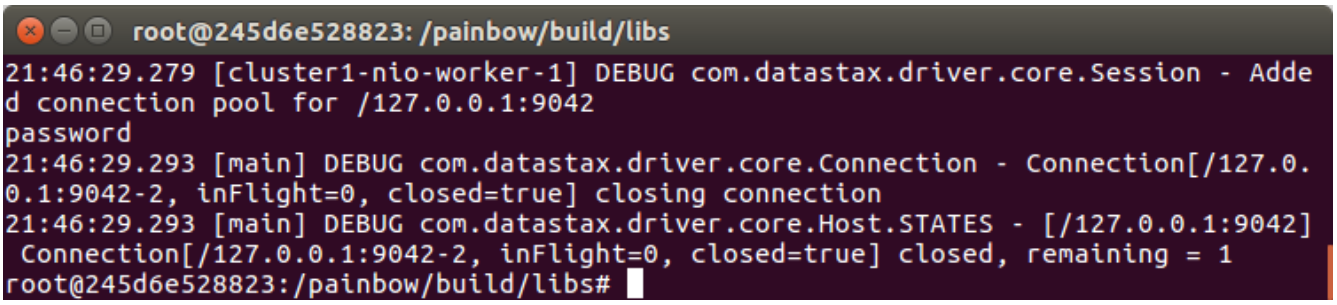
Compute the MD5 hash of “password”



Result of MD5 hash of “password” followed by decrypting the resulting hash



Result of decrypting the hash for “password”



Detailed Explanation of Dockerfile (comments begin with # and are not part of the dockerfile)

```
# I used ubuntu 16.04 because I first built painbow on my laptop which is ubuntu 16.04
FROM ubuntu:16.04

# The following commands install software necessary for painbow to be built and run
RUN apt-get -y update                   # Update the package list
&& apt-get install -y gradle            # Install packages necessary for building painbow
&& apt-get install -y wget
&& apt-get install -y curl
&& apt-get install -y git
&& apt-get install -y default-jdk       # Install Java
#The next three commands install cassandra and are from cassandra's website
&& echo "deb http://www.apache.org/dist/cassandra/debian 310x main" | tee -a
/etc/apt/sources.list.d/cassandra.sources.list
&& curl https://www.apache.org/dist/cassandra/KEYS | apt-key add -
&& apt-get update && apt-get -y install cassandra

# The next section of RUN commands builds painbow from the github repo
RUN git clone http://github.com/mcandre/painbow.git       # Clone from github
&& cd painbow
&& gradle build         # Build painbow
&& cd build/libs        # go to directory containing painbow.jar created by build
# download dependencies to the folder containing painbow.jar
&& wget http://central.maven.org/maven2/com/offbytwo/docopt/0.6.0.20150202/docopt-
0.6.0.20150202.jar
&& wget http://central.maven.org/maven2/commons-codec/commons-codec/1.10/commons-codec-
1.10.jar

# Command executed when the container is run
CMD cd painbow/build/libs;              # go to the painbow directory
# define an alias so the user can use painbow as a command. This makes it seem like painbow is
actually installed even though it is not
echo 'alias painbow="java -cp painbow.jar:commons-codec-1.10.jar:docopt-
0.6.0.20150202.jar:/usr/share/cassandra/lib/* us.yellosoft.painbow.Painbow $1"' >> ~root/.bashrc;
# start cassandra
service cassandra start;
# create a shell for the user
/bin/bash
```

Usage:
Build with 'docker build -t painbow-final .'
Run with 'docker run -it painbow-final'

Wait a few seconds to give cassandra time to start
Inside the shell, run 'painbow –migrate'
Run other painbow commands from the github page

3.5 Enhanced Password Helper (4 hours)

I also reworked the user interface of the APK. To keep the functionality from P5, I made the main activity a sort of menu where the user can select whether to use the word lists or strength estimator. For education, I added another activity which contains clickable hyperlinks to articles on password security.

Password Estimation

For the password strength estimator, I added Nbvcxz to my android project in the gradle file. To use Nbvcxz, I also had to download and install a newer version of gradle

Added Nbvcxz to gradle file

```
apply plugin: 'com.android.application'
apply plugin: 'maven'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.example.frodo.passwordsecure"
        minSdkVersion 24
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.1.0'
    compile 'me.gosimple:nbvcxz:+'
    testCompile 'junit:junit:4.12'
}
```

Changed gradle version from 2.14 to 3.5 in the gradle wraper

AndroidManifest.xml × gradle-wrapper.properties ×

```
#Mon Dec 28 10:00:20 PST 2015
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-3.5-all.zip
```

The code uses Nbvcxz for password strength estimation. It gives three outputs to the user: the password's entropy, a warning (how common the password appears to be), and suggestions (things like captitalization doesn't help security).

```
public void checkPasswordPressed(View view)
{
     passRater = new Nbvcxz();
     Result result = passRater.estimate(password.getText().toString());
     String feedback = "";
     feedback += "Entropy: "+Double.toString(result.getEntropy())+"\n\n";
     if(result.getFeedback().getWarning() != null)
          feedback += "Warnings: "+ result.getFeedback().getWarning()+"\n\n";
     if(result.getFeedback().getSuggestion() != null)
          feedback += "Suggestions: "+result.getFeedback().getSuggestion().toString();
     output.setText(feedback);
}
```

Learning about passwords

Layout xml file contains four text views whose text is set to strings from strings.xml. One of these textviews is shown below:

```
<TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_below="@+id/title"
     android:id="@+id/schneier"
     android:text="@string/schneierlink"/>
```

The strings.xml file contains html link tags (escaped with \) which are displayed in the text views.

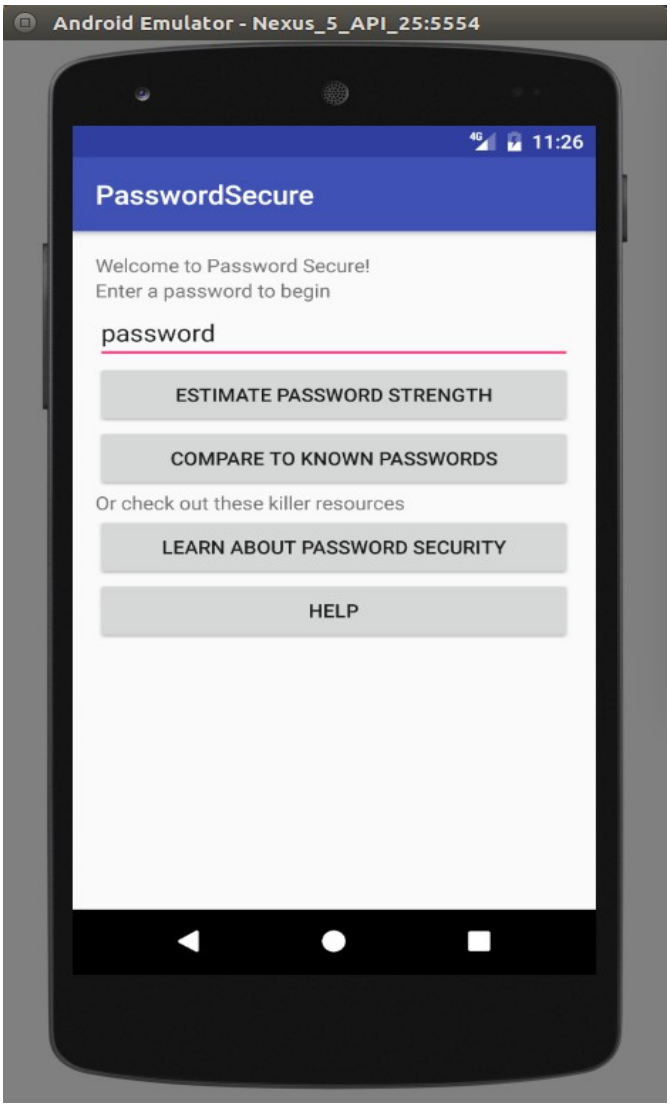
```
<string name="schneierlink">\<a
href='https://www.schneier.com/blog/archives/2014/03/choosing_secure_1.html'>Password
Choosing Tips</a></string>
<string name="cmulink">\<a
href='https://www.cs.cmu.edu/~help/security/choosing_passwords.html'>More Password Choosing
Tips</a></string>
<string name="arstechnicalink">\<a href='https://arstechnica.com/security/2013/05/how-
crackers-make-minced-meat-out-of-your-passwords/'>How Passwords Are Cracked</a></string>
<string name="googlelink">\<a href='http://google.com/search?q=password+security'>Search
google</a></string>
```

In order to make the links clickable, the java file calls setMovementMethod on each of the text views:

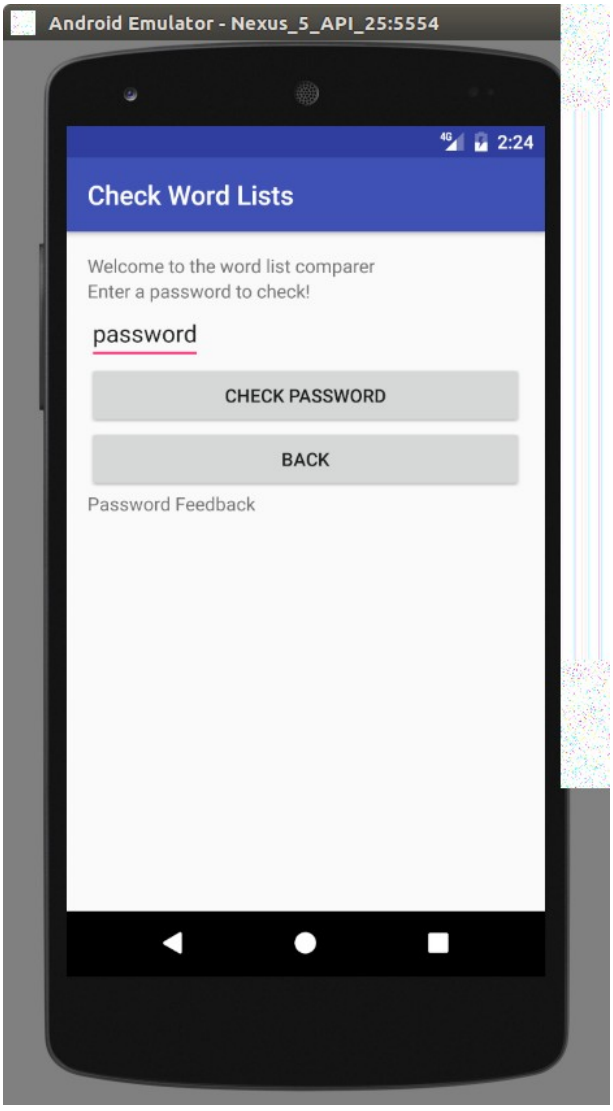
```
TextView google;
TextView schneier;
TextView cmu;
TextView arstechnica;
@Override
protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_learn_stuff);
     google = (TextView) findViewById(R.id.google);
     google.setMovementMethod(LinkMovementMethod.getInstance());
     schneier = (TextView) findViewById(R.id.schneier);
     schneier.setMovementMethod(LinkMovementMethod.getInstance());
     cmu = (TextView) findViewById(R.id.cmu);
     cmu.setMovementMethod(LinkMovementMethod.getInstance());
     arstechnica = (TextView) findViewById(R.id.arstechnica);
     arstechnica.setMovementMethod(LinkMovementMethod.getInstance());
}
```

Screenshots

Home screen of the app

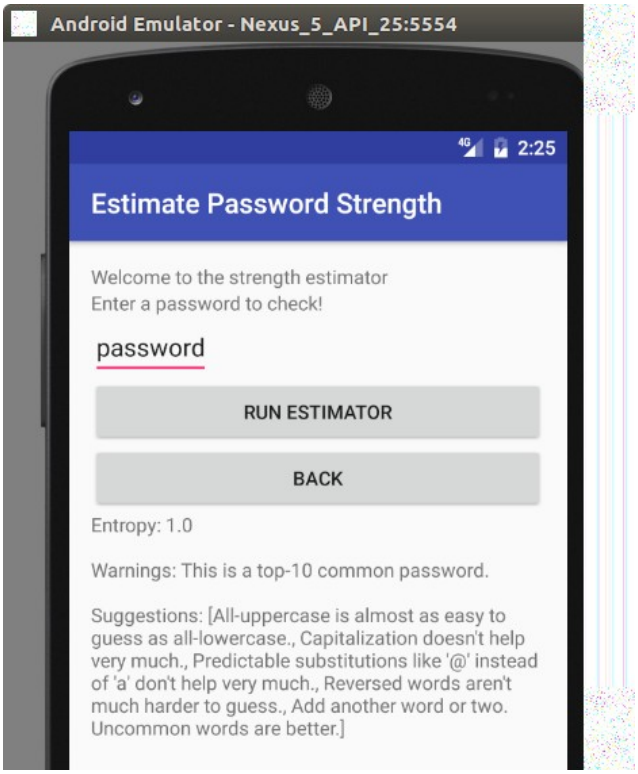


Compare to known passwords from P5

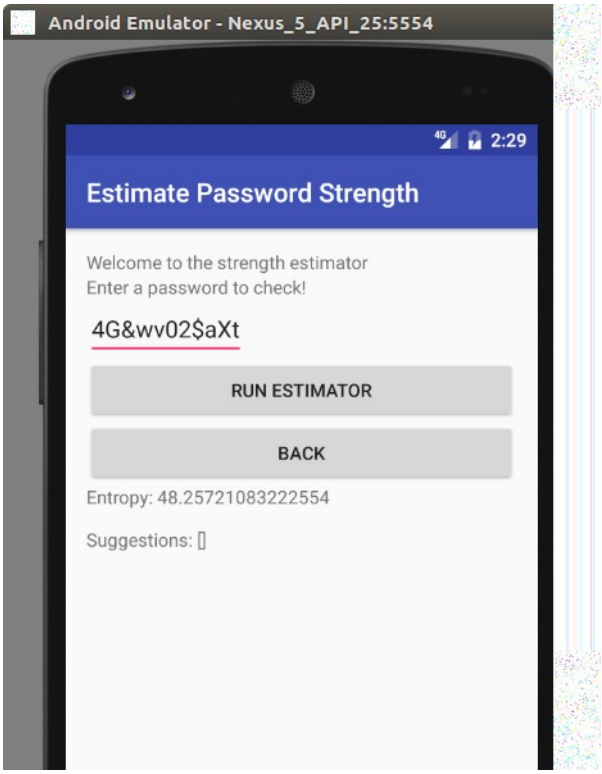


Password strength estimator

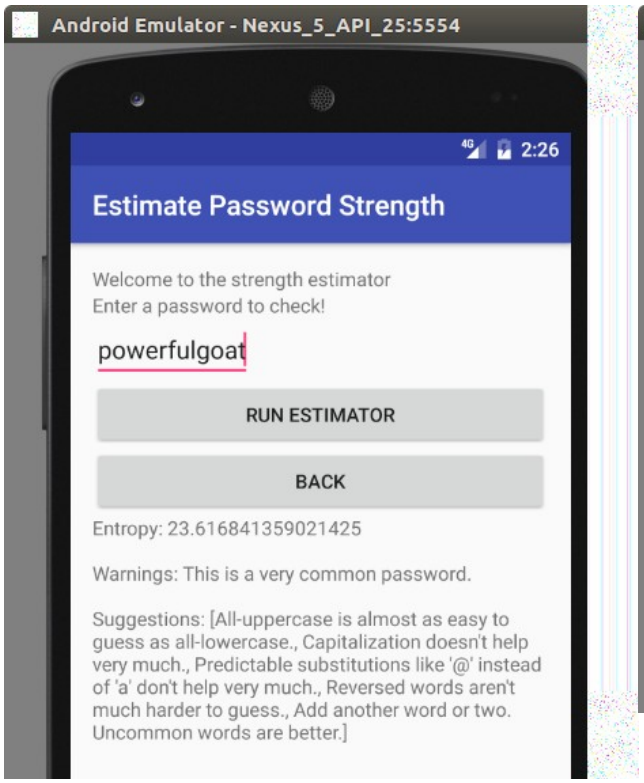
A very common password



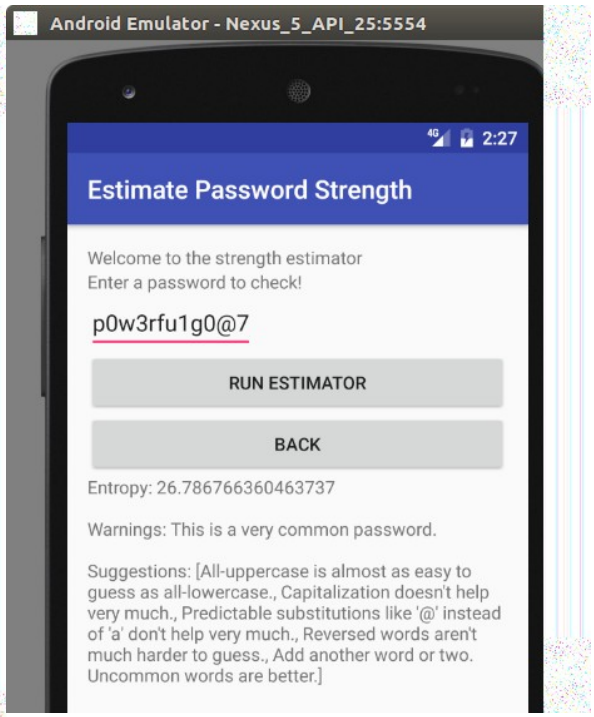
A strong password



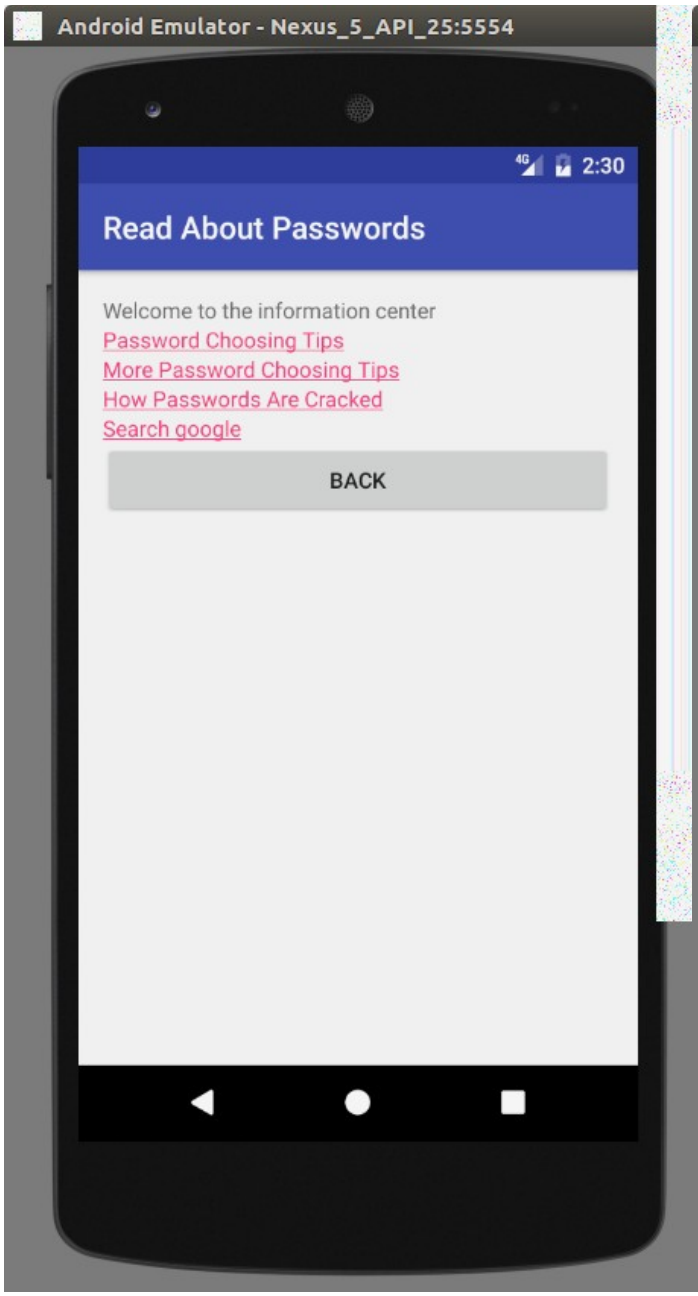
A bad password



Making substitutions doesn’t help much



Learn about passwords activity



Clicking a link opens it in a web browser